Featured Graph Coarsening with Similarity Guarantees

Manoj Kumar [1] **Anurag Sharma** [2] **Shashwat Saxena** [1] **Sandeep Kumar** [1 3 4]

Abstract

Graph coarsening is a dimensionality reduction technique that aims to learn a smaller-tractable graph while preserving the properties of the original input graph. However, many real-world graphs also have features or contexts associated with each node. The existing graph coarsening methods do not consider the node features and rely solely on a graph matrix(e.g., adjacency and Laplacian) to coarsen graphs. However, some recent deep learning-based graph coarsening methods are designed for specific tasks considering both node features and graph matrix. In this paper, we introduce a novel optimization-based framework for graph coarsening that takes both the graph matrix and the node features as the input and jointly learns the coarsened graph matrix and the coarsened feature matrix while ensuring desired properties. To the best of our knowledge, this is the first work that guarantees that the learned coarsened graph is $\epsilon \in [0, 1)$ similar to the original graph. Extensive experiments with both real and synthetic benchmark datasets elucidate the proposed framework's efficacy and applicability for numerous graph-based applications, including graph clustering, node classification, stochastic block model identification, and graph summarization.

1. Introduction

Graph-based approaches with big data and machine learning are one of the strongest driving forces of the current research frontiers, creating new possibilities in various domains, from social networks to drug discovery and from finance to material science studies. Large-scale graphs are becoming increasingly common, which is exciting since more data implies more knowledge and more training sets for learning algorithms. However, the graph data size is the real bottleneck; handling large graph data involves considerable computational hurdles to process, extract, and analyze. Therefore, graph dimensionality reduction techniques are needed.

Graph coarsening or graph summarization is a promising direction for scaling up graph-based machine-learning approaches by simplifying large graphs. Coarsening aims to summarize a very large graph into a smaller and tractable graph while preserving the properties of the originally given graph. The core idea of coarsening comes from the algebraic multi-grid literature (Ruge & Stüben, 1987). Coarsening methods have been applied in various applications like graph partitioning (Hendrickson et al., 1995; Karypis & Kumar, 1998; Kushnir et al., 2006; Dhillon et al., 2007), graph summarization (Liu et al., 2018), machine learning (Lafon & Lee, 2006; Gavish et al., 2010; Shuman et al., 2015), and scientific computing(Chen et al., 2022; Hackbusch, 2013; Ruge & Stüben, 1987; Briggs et al., 2000). The recent work in (Loukas, 2019) developed a set of frameworks for graph matrix coarsening, preserving spectral and cut guarantees, but do not take node features into account. There are some recent deep learning-based approaches that do take node features into account (Cai et al., 2021; Ying et al., 2018; Ma & Chen, 2020).

Furthermore, many real-world graphs also have features associated with each graph, e.g., node feature or edge feature (Kipf & Welling, 2017; Zügner & Günnemann, 2019; Wang et al., 2019). Existing graph coarsening methods are not designed to consider features of nodes and rely solely on structural information e.g., adjacency matrix to simplify graphs that may not be suitable for downstream tasks that require node features. Furthermore, these methods also lack a formal guarantee on the properties of the original graph and features being preserved in the coarsened graph and coarsened features. However, there are some recent deep learning-based graph coarsening technique that considers feature matrix and Laplacian matrix both designed for a particular task only, e.g., optimal transport coarsening(Ma & Chen, 2021) for graph classification, GCOND (Jin et al.,

[1]Department of Electrical Engineering, IIT Delhi, New Delhi, India [2]Department of Mathematics and Computing, IIT Delhi, New Delhi, India [3]Bharti School Of Telecommunications Technology Management, IIT Delhi, New Delhi, India [4]Yardi School of Artificial Intelligence, IIT Delhi, New Delhi, India. Correspondence to: Manoj Kumar <eez208646@iitd.ac.in>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2021), SCAL (Huang et al., 2021) for node classification, DosCond(Jin et al., 2022) for node and graph classification both.

We introduce a novel optimization-based framework lying at the unification of graph learning (Kumar et al., 2020; 2019) and dimensionality reduction (Qiu et al., 2017; Zhu et al., 2017) for coarsening graph data, named as featured graph coarsening (FGC). It takes both the graph matrix and the node features as the input and learns the coarsened graph matrix and the coarsened feature matrix jointly while ensuring desired properties. The proposed optimization formulation is a multi-block non-convex optimization problem, which is solved efficiently by leveraging block majorization-minimization, log determinant, Dirichlet energy, and regularization frameworks. The developed algorithm is provably convergent and enforces the desired properties in the learned coarsened graph. To the best of our knowledge, the proposed method is the first coarsening method that guarantees that the original graph and coarsened graph are ϵ-similar for $\epsilon \in [0, 1)$. Extensive experiments elucidate the efficacy of the proposed framework for real-world applications.

Furthermore, we have also performed the downstream task i.e. node classification on small and large datasets both to elucidate the efficacy of graph coarsening. We have trained graph neural network (GNN) using the coarsened graph and assign the label of original graph. We have compared the node classification accuracy and computational time required to perform coarsening and classification against the most recent state of the art methods like GCOND (Jin et al., 2021), SCAL (Huang et al., 2021).

Remark: It is important here to highlight the distinction between (i) Clustering (Ng et al., 2001; Dhillon et al., 2007) (ii) Community detection (Fortunato, 2010), and (iii) Graph Coarsening methods (Loukas & Vandergheynst, 2018). Given a set of data points, the clustering and community detection algorithm aim to segregate groups with similar traits and assign them into clusters. For community detection, the data points are nodes of a given network. But these methods do not answer how these groups are related to each other. On the other hand coarsening segregates groups with similar traits and assigns them into supernodes, in addition, it also establishes how these supernodes are related to each other. It learns the graph of the supernodes, the edge weights, and finally the effective feature of each supernode. The scope of coarsening is wider than the aforementioned methods.

Notation In terms of notation, lower case (bold) letters denote scalars (vectors) and upper case letters denote matrices. The (i, j)-th entry of a matrix X is denoted by X_{ij}. $X^\dagger$ and $X^\top$ denote the pseudo inverse and transpose of matrix X, respectively. X_i and $[X^T]_j$ denote the i-th column and j-th row of matrix X. $\langle X_i, X_j \rangle = X_i^T X_j$ denotes the inner product of two vectors. The all-zero and all-one vectors or matrices of appropriate sizes are denoted by $\mathbf{0}$ and $\mathbf{1}$, respectively. The $\|X\|_1$, $\|X\|_F$, $\|X\|_{1,2}$ denote the ℓ_1-norm, Frobenius norm and $\ell_{1,2}$-norm of X, respectively. $\det(X)$ is defined as the generalized determinant of a positive definite matrix X.

2. Background

In this section, we review the basics of graph and graph coarsening, the spectral similarity of the graph matrices, and the ϵ-similarity of graph matrices and feature matrices.

2.1. Graph

A graph with features is denoted by $\mathcal{G} = (V, E, W, X)$ where $V = \{v^1, v^2, ..., v^p\}$ is the vertex set, $E \subseteq V \times V$ is the edge set and W is the adjacency (weight) matrix. We consider a simple undirected graph without self-loop: $W_{ij} > 0$, if $(i, j) \in E$ and $W_{ij} = 0$, if $(i, j) \notin E$. Finally, $X \in \mathbb{R}^{p \times n} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p]^\top$ is the feature matrix, where each row vector $\mathbf{x}_i \in \mathbb{R}^n$ is the feature vector associated with one of p nodes of the graph $\mathcal{G}$. Thus, each of the n columns of X can be seen as a signal on the same graph. Graphs are conveniently represented by some matrix, such as Laplacian and adjacency graph matrices, whose positive entries correspond to edges in the graph.

A matrix $L \in \mathbb{R}^{p \times p}$ is a combinatorial graph Laplacian matrix if it belongs to the following set:

$$\mathcal{S}_L = \{L_{ij} = L_{ji} \leq 0 \text{ for } i \neq j; L_{ii} = -\sum_{j \neq i} L_{ij}\}. \quad (1)$$

The W and the L are related as follows: $W_{ij} = -L_{ij}$ for $i \neq j$ and $W_{ij} = 0$ for $i = j$. Both L and W represent the same graph, however, they have very different mathematical properties. The Laplacian matrix L is a symmetric, positive semidefinite matrix with zero row sum. The non-zero entries of the matrix encode positive edge weights as $-L_{ij}$ and $L_{ij} = 0$ implies no connectivity between vertices i and j. The importance of the graph Laplacian matrix has been well recognized as a tool for embedding, manifold learning, spectral sparsification, clustering, and semi-supervised learning. Owing to these properties, Laplacian matrix representation is more desirable for building graph-based algorithms.

2.2. Graph Coarsening

Given an original graph $\mathcal{G} = (V, E, W, X)$ with p nodes, the goal of graph coarsening is to construct an appropriate "smaller" or coarsened graph $\mathcal{G}_c = (\tilde{V}, \tilde{E}, \tilde{W}, \tilde{X})$ with $k << p$ nodes, such that $\mathcal{G}_c$ and $\mathcal{G}$ are similar in some sense. Every node $\tilde{v}^j \in \tilde{V}$, where $j = 1, 2, ...k$, of the smaller graph with reference to the nodes of the larger graph is termed as a "super-node". In coarsening, we define a linear mapping $\pi : V \to \tilde{V}$ that maps a set of nodes in $\mathcal{G}$ having similar properties to a super-node in $\mathcal{G}_c$ i.e. for any super-

node $\tilde{v} \in \tilde{V}$, all nodes $\pi^{-1}(\tilde{v}) \subset V$ have similar properties. Furthermore, the features of the super-node, $\tilde{v}$, should be based on the features of nodes $\pi^{-1}(\tilde{v}) \subset V$ in $\mathcal{G}$, and the edge weights of the coarse graph, $\tilde{W}$, should depend on the original graph's weights as well as the coarsened graph's features.

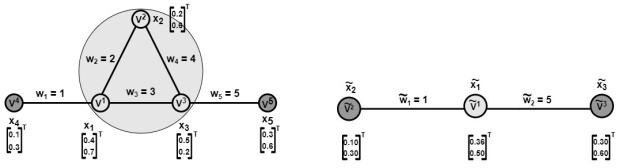
Let $P \in \mathbb{R}_+^{k\times p}$ be the coarsening matrix which is a linear map from $\pi : V \to \tilde{V}$ such that $\tilde{X} = PX$. Each non-zero entry of P i.e. $[P]_{ij}$, indicate the j-th node of $\mathcal{G}$ is mapped to i-th super node of $\mathcal{G}_c$. For example, non-zero elements of j-th row, i.e., $\mathbf{p}_j$ corresponds to the following nodes set $\pi^{-1}(\tilde{v}_j) \in V$. The rows of P will be pairwise orthogonal if any node in V is mapped to only a single super-node in $\tilde{V}_c$. This means that the grouping via super-node is disjoint. Let the Laplacian matrices of $\mathcal{G}$ and $\mathcal{G}_c$ be $L \in \mathbb{R}^{p\times p}$ and $L_c \in \mathbb{R}^{k\times k}$, respectively. The Laplacian matrices L, L_c, feature matrices X, $\tilde{X}$ and the coarsening matrix P together satisfy the following properties(Loukas, 2019):

$$L_c = C^T L C, \quad \tilde{X} = PX, \quad X = P^\dagger \tilde{X} = C\tilde{X} \tag{2}$$

where $C \in \mathbb{R}^{p\times k}$ is the tall matrix which is the pseudo inverse of P and known as the loading matrix. The non-zero elements of C, i.e., $C_{ij} > 0$ implies that the i-th node of $\mathcal{G}$ is mapped to the j-th supernode of $\mathcal{G}_c$. The loading matrix $C \in \mathbb{R}_+^{p\times k}$ belongs to the following set:

$$\mathcal{C} = \Big\{ C \geq 0 |\ \langle C_i, C_j\rangle = 0 \ \forall\ i \neq j, \quad \langle C_i, C_i\rangle = d_i, \quad \|C_i\|_0 \geq 1 \ and\ \big\|[C^T]_i\big\|_0 = 1 \Big\} \tag{3}$$

where C_i and C_j represent i-th and j-th column of loading matrix C and they are orthogonal to each other, $[C^T]_i$ represents the i-th row of loading matrix C, and d_i's are positive quantities. The C matrix has k columns and p rows. Also, in each row of the loading matrix C, there is only one non-zero entry and that entry is 1 which implies that $C \cdot \mathbf{1}_k = \mathbf{1}_p$, where $\mathbf{1}_k$ and $\mathbf{1}_p$ are vectors having all entry 1 and having the size of k and p respectively. This implies that $C^T C$ will be a diagonal matrix with d_i as the ith diagonal element.



(a) Original graph $\mathcal{G}$ (b) Coarsened graph $\mathcal{G}_c$

Figure 1: Toy example: Featured graph coarsening.

In the toy example, nodes (v^1, v^2, v^3) of $\mathcal{G}$ are mapped into super-node $\tilde{v}^1$ of $\mathcal{G}_c$. The coarsening matrix P and the loading matrix C are

$$P = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad C = P^\dagger = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

From the coarsened dimension of $k \times k$ one can go back to the original dimension, i.e., $p \times p$ by computing the lifted Laplacian matrix defined as $L_l = P^T L_c P$(Loukas & Vandergheynst, 2018).

2.3. Preserving properties of $\mathcal{G}$ in $\mathcal{G}_c$

The coarsened graph $\mathcal{G}_c(L_c, \tilde{X})$ should be learned such that the properties of $\mathcal{G}$ and $\mathcal{G}_c$ are similar. Some widely used notions of similarities are :

Definition 1. ***Spectral similarity****(Loukas & Vandergheynst, 2018; Loukas, 2019) The spectral similarity is shown by calculating relative eigen error (REE), defined as* $REE = \frac{1}{m}\sum_{i=1}^{m} \frac{|\tilde{\lambda}_i - \lambda_i|}{\lambda_i}$*, where* λ_i *and* $\tilde{\lambda}_i$ *are the top* m *eigenvalues corresponding to the original graph Laplacian matrix* L *and coarsened graph Laplacian matrix* L_c *respectively and* m *is the count of eigen value.*

Definition 2. *Let* L *be original Laplacian matrix and* L_l *be the lifted Laplacian matrix,then the reconstruction error(RE) (Liu et al., 2018) is defined as* $RE = \|L - L_l\|_F^2$.

Definition 3. ***Hyperbolic Error(HE)*** *(Bravo Hermsdorff & Gunderson, 2019) The hyperbolic error between original Laplacian matrix* L *and lifted Laplacian matrix* L_l *is defined as* $HE = arccosh(1 + \frac{\|(L-L_l)X\|_F^2 \|X\|_F^2}{2tr(X^T L X) tr(X^T L_l X)})$.

The REE value indicates that how well the eigen properties of the original graph $\mathcal{G}$ is preserved in the coarsened graph $\mathcal{G}_c$. The RE and HE values indicate how well we can recover the original graph from the coarsened graph. For a good coarsening algorithm lower values of these quantities are desired.

3. Featured Graph Coarsening (FGC)

The existing graph coarsening or summarization methods are not designed to consider the node features and solely rely on the graph matrix for learning a simpler graph (Loukas & Vandergheynst, 2018; Loukas, 2019; Bravo Hermsdorff & Gunderson, 2019; Purohit et al., 2014; Chen et al., 2022; Hajiabadi et al., 2021; Riondato et al., 2017; Kang et al., 2022; Ko et al., 2020), and thus, not suitable for graph machine learning applications. For example, many real-world graph data satisfy certain properties, e.g., homophily assumption and smoothness (Wang et al., 2021; Kalofolias, 2016), that if two nodes are connected with stronger weights, then the features corresponding to these nodes should be similar. Thus, if the original graph satisfies any property, then that property should translate to the coarsened graph data. Current methods can only ensure spectral properties which satisfy the property of the graph matrix but not the features (Loukas & Vandergheynst, 2018; Loukas, 2019; Chen et al., 2022).

The aforementioned discussion suggests the following graph coarsening method (i) should consider jointly both the graph

matrix L and the node feature X of the original graph and (ii) to ensure the desired specific properties on coarsened graph data, such as smoothness and homophily, the L_c and $\tilde{X}$ should be learned jointly depending on each other. We approach this problem at the unification of dimensionality reduction (Qiu et al., 2017; Zhu et al., 2017) and graph learning (Kumar et al., 2020; 2019), where we solve these problems jointly, first we reduce the dimensionality and then learn a suitable graph on the reduced dimensional data. We propose a unique optimization-based framework that uses both the features X and Laplacian matrix L of the original graph to learn loading matrix C and coarsened graph's features $\tilde{X}$, jointly. Next we briefly discuss how to learn graphs with features, and then we propose our formulation.

3.1. Graph learning from data

Let $X = [\mathbf{x}_1, ..., \mathbf{x}_p]^T$, where $\mathbf{x}_i$ is $n-$dimensional feature vector associated with $i-$th node of an undirected graph. In the context of modeling signals or features with graphs, the widely used assumption is that the signal residing on the graph changes smoothly between connected nodes (Kalofolias, 2016). The Dirichlet energy (DE) is used for quantifying the smoothness of the graph signals which is defined by using graph Laplacian matrix $L \in \mathcal{S}_L$ matrix and the feature vector as follows:

$$\text{DE}(L, X) = \text{tr}(X^T L X) = -\sum L_{ij} \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2. \quad (4)$$

The lower value of Dirichlet energy indicates a more desirable configuration (Wang et al., 2021). Smooth graph signal methods are an extremely popular family of approaches for a variety of applications in machine learning and related domains (Dong et al., 2016). When only the feature matrix $X = [\mathbf{x}_1, ..., \mathbf{x}_p]^T$, associated with an undirected graph is given then a suitable graph satisfying the smoothness property can be obtained by solving the following optimization problem:

$$\min_{L \in \mathcal{S}_L} -\gamma \log(\det(L + J)) + \text{tr}(X^T L X) + \alpha h(L) \quad (5)$$

where $L \in \mathbb{R}^{p \times p}$ denotes the desired graph matrix, $\mathcal{S}_L$ is the set of Laplacian matrix (1), $h(\cdot)$ is the regularization term, and $\alpha > 0$ is the regularization parameter, and $J = \frac{1}{p}\mathbf{1}_{p \times p}$ is a constant matrix whose each element is equal to $\frac{1}{p}$. The rank of L is $p - 1$ for connected graph matrix having p nodes(Chung & Graham, 1997), adding J to L makes $L + J$ a full rank matrix without altering the row and column space of the matrix L(Kumar et al., 2020; Kalofolias, 2016).

When the data is Gaussian distributed $X \sim \mathcal{N}(\mathbf{0}, L^\dagger)$, optimization in (5) also corresponds to the penalized maximum likelihood estimation of the inverse covariance (precision) matrix also known as Gaussian Markov random field (GMRF) for $\gamma = 1$ (Ying et al., 2020). The graph $\mathcal{G}$ inferred from L and the random vector X follows the Markov property, meaning $L_{ij} \neq 0 \iff \{i, j\} \in E \ \forall i \neq j$ implies $\mathbf{x}_i$ and $\mathbf{x}_j$ are conditionally dependent given the rest. Furthermore, in a more general setting with non-Gaussian distribution, (5) can be related to the log-determinant Bregman divergence regularized optimization problem, which ensures nice properties on the learned graph matrix, e.g., connectedness and full rankness.

3.2. Proposed Featured Graph Coarsening Formulation

The proposed formulation of FGC is

$$\begin{aligned} \min_{L_c, \tilde{X}, C} & -\gamma \log(\det(L_c + J)) + \text{tr}(\tilde{X}^T L_c \tilde{X}) \quad (6) \\ & + \beta h(L_c) + \frac{\lambda}{2} g(C) \\ \text{s.t.} \quad & C \geq 0,\ L_c = C^T L C,\ X = C\tilde{X},\ L_c \in \mathcal{S}_L, \end{aligned}$$

where L and X are the given Laplacian and feature matrix of a large connected graph, and $\tilde{X} \in \mathbb{R}^{k \times n}$ and $L_c \in \mathbb{R}^{k \times k}$ are the feature matrix and the Laplacian matrix of the learned coarsened graph, respectively, $C \in \mathbb{R}^{p \times k}$ is the loading matrix, $h(\cdot)$ and $g(\cdot)$ are the regularization functions for L_c and the loading matrix C, while $\beta > 0$ and $\lambda > 0$ are the regularization parameters. Fundamentally, the proposed formulation (6) aims to learn the coarsened graph matrix L_c, the loading matrix C, and the feature matrix $\tilde{X}$, jointly. This constraint $X = C\tilde{X}$ coarsens the feature matrix of larger graph $X \in \mathbb{R}^{p \times n}$ to a smaller graph's feature matrix $\tilde{X} \in \mathbb{R}^{k \times n}$. Next, the first two-term of the objective function is the graph learning term, where the $\log\det(\cdot)$ term ensures the coarsened graph is connected while the second term imposes the smoothness property on the coarsened graph, and finally third and fourth term act as a regularizer. The regularizer $g(C)$ ensures the mapping such that one node $v^i \in V$ does not get mapped to two different super-nodes $\tilde{v}^j, \tilde{v}^k \in \tilde{V}$ and mapping of nodes to super-nodes be balanced such that not all or majority of nodes get mapped to the same super-node. This simply implies that only one element of each row of C be non-zero and the columns of C be sparse. An $\ell_{1,2}$-based group penalty is suggested to enforce such structure (Yuan & Lin, 2006; Ming et al., 2019).

3.3. Featured Graph Coarsening (FGC) Algorithm

Next by using $L_c = C^T L C$ and introducing a quadratic penalty for $X = C\tilde{X}$ we aim to solve the following optimization problem:

$$\begin{aligned} \min_{\tilde{X}, C} & -\gamma \log\det(C^T L C + J) + \text{tr}(\tilde{X}^T C^T L C \tilde{X}) \quad (7) \\ & + \frac{\alpha}{2} ||C\tilde{X} - X||_F^2 + \frac{\lambda}{2} \|C^T\|_{1,2}^2 \\ \text{s.t.} \quad & \mathcal{S}_C = \left\{C \geq 0 |\ \|[C^T]_i\|_2^2 \leq 1\ \forall\, i = 1, .., p\right\} \end{aligned}$$

where $\|C^T\|_{1,2}^2 = \sum_{i=1}^{p} \|[C^T]_i\|_1^2 = \sum_{i=1}^{p} (\sum_{j=1}^{k} C_{ij})^2$ is the $\ell_{1,2}$ norm of C^T which ensures group sparsity in the resultant C matrix and $[C^T]_i$ is the i-th row of matrix C. For a high value of λ, the loading matrix is observed to be orthogonal, more details are given in remark 2. The quadratic relaxation $\frac{\alpha}{2}||C\tilde{X} - X||_F^2$ keeps $C\tilde{X}$ close to X instead of solving the constraint. Note that this relaxation can be made tight by using sufficiently large or iteratively increasing α.

3.3.1. Some properties of C^TLC matrix

Before we proceed with the algorithm development, some of the properties and intermediary Lemmas regarding C^TLC are presented below.

Lemma 1. *If L is the Laplacian matrix for a connected graph with p nodes, and C is the loading matrix such that $C \in \mathbb{R}_+^{p\times k}$ and $C \in \mathcal{C}$ as in (3), then the coarsened graph matrix $L_c = C^TLC$ is a connected graph Laplacian matrix with k nodes.*

Lemma 2. *If L be the Laplacian matrix for a connected graph with p nodes, $C \in \mathcal{C}$ be the loading matrix, and $J = \frac{1}{k}\mathbf{I}_{k\times k}$ is a constant matrix whose each element is equal to $\frac{1}{k}$. The function $f(C) = -\gamma log\ det(C^TLC + J)$ is a convex function with respect to the loading matrix C.*

We provide the sketches of the proofs of Lemma 1 and Lemma 2 here. Please refer to the appendix A.1 and A.2 for detailed proof.

Proof sketch of Lemma 1. Using Cholesky decomposition and (3), we have proved that $L_c = C^TLC$ is the symmetric positive semi-definite matrix. Also, $C \cdot \mathbf{u}_k = \mathbf{u}_p$ if and only if $\mathbf{u}_k = t\mathbf{1}_k$ and $\mathbf{u}_p = t\mathbf{1}_p$ and $C \cdot \mathbf{u}_k \neq \mathbf{u}_p\ \forall\ \mathbf{u}_k \neq t\mathbf{1}_k$ and $\mathbf{u}_p \neq t\mathbf{1}_p$ where $t \in \mathbb{R}$ which implies that $C \cdot \mathbf{u}_k = \mathbf{u}_p$ holds only for a constant vector $\mathbf{u}_k$. And thus, $C^TLC \cdot \mathbf{u}_k = \mathbf{0}_k$, for constant vector $\mathbf{u}_k$. This concludes that the constant vector is the only eigenvector spanning the nullspace of L_c, concluding that the rank of C^TLC is $k-1$.

Proof sketch of Lemma 2. Since C^TLC+J is a symmetric positive definite matrix and using Cholesky decomposition it can be written as $C^TLC = Y^TY$ where we establish that $Y = L^{\frac{1}{2}}C + \frac{1}{\sqrt{pk}}\mathbf{1}_{p\times k}$. And $-\gamma$log det(Y^TY) is a convex function in Y. Furthermore, C is an affine transformation of Y implying $-\gamma$log det$(C^TLC + J)$ is also a convex function in C.

3.4. BSUM Framework

The resulting optimization problem in (7) is a multi-block non-convex. We develop efficient optimization methods based on the block Majorization Minimization framework. Suppose we have the following optimization problem

$$\underset{\mathbf{x}}{\text{minimize}}\ f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{x} \in \mathcal{X}, \tag{8}$$

where $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$, with $\mathbf{x}_i \in \mathcal{X}_i$, $\mathcal{X} = \prod_{i=1,2} \mathcal{X}_i$ is a closed convex set, and $f : \mathcal{X} \to \mathbb{R}$ is a continuous function. At the t-th iteration, block $\mathbf{x}_1$ is updated in a cyclic order by solving:

$$\underset{\mathbf{x}_1}{\text{minimize}}\ g_1\left(\mathbf{x}_1|\mathbf{x}_2^{(t)}\right) \quad \text{subject to} \quad \mathbf{x}_1 \in \mathcal{X}_1,$$

where a continuous function $g_1\left(\mathbf{x}_1|\mathbf{x}_2^{(t)}\right)$ is a majorization function of $f(\mathbf{x}_1)$ at $\mathbf{x}_2^{(t)}$ satisfying

$$g_1\left(\mathbf{x}_1^{(t)}|\mathbf{x}_2^{(t)}\right) = f\left(\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}\right) \tag{9a}$$

$$g_1\left(\mathbf{x}_1|\mathbf{x}_2^{(t)}\right) \geq f\left(\mathbf{x}_1, \mathbf{x}_2^{(t)}\right),\ \forall\ \mathbf{x}_1 \in \mathcal{X}_1, \tag{9b}$$

$$\nabla g_1\left(\mathbf{x}_1; \mathbf{d}_1|x_2^{(t)}\right)|_{\mathbf{x}_1=\mathbf{x}_1^{(t)}} = \nabla f\left(\mathbf{x}_1^{(t)}, \mathbf{x}_2^{(t)}; \mathbf{d}\right), \tag{9c}$$

$$\text{such that } \mathbf{x}_1^{(t)} + \mathbf{d}_1 \in \mathcal{X}_1,$$

where $\nabla f(\mathbf{x}, \mathbf{d})$ stands for the directional derivative at $\mathbf{x}$ along $\mathbf{d} = (\mathbf{d}_1, 0)$ (Razaviyayn et al., 2012; Sun et al., 2017). Similarly we can find the surrogate function $g_2\left(\mathbf{x}_2|\mathbf{x}_1^{(t)}\right)$ at the t-th iteration. If the surrogate functions g_i is properly chosen, then the solution to (9a) could be easier to obtain than solving (8) directly.
Collecting the variables as $(C \in \mathbb{R}_+^{p\times k}, \tilde{X} \in \mathbb{R}^{k\times n})$, we develop a block MM-based algorithm which updates one variable at a time while keeping the other fixed.

3.5. Update of C

Treating C as a variable and fixing $\tilde{X}$, we obtain the following sub-problem for C:

$$\min_{C\in\mathcal{S}_c} f(C) = -\gamma \log\det(C^TLC + J) + \frac{\lambda}{2}\|C^T\|_{1,2}^2 \tag{10}$$
$$+\frac{\alpha}{2}\|C\tilde{X} - X\|_F^2 + \text{tr}(\tilde{X}^TC^TLC\tilde{X})$$

The function $f(C)$ in (10) is a strictly convex function. This can be established from Lemmas 1 and 2 and using the property of norm, and trace operators, see the appendix A.5 for detailed proof. The set $\mathcal{S}_C$ is a closed and convex set, thus (10) is a strongly convex optimization problem, but there does not exist a closed-form solution to it. To get closed-form updates we will use the MM framework (Beck & Pan, 2018; Razaviyayn et al., 2012; Sun et al., 2017).

By using the first-order Taylor series approximation, a majorized function for $f(C)$ at $C^{(t)}$ can be obtained as:

$$g(C|C^{(t)}) = f(C^{(t)}) + (C - C^{(t)})\nabla f(C^{(t)}) + \frac{L}{2}||C - C^{(t)}||^2 \tag{11}$$

where $f(C)$ is $L-$Lipschitz continuous gradient function $L = \max(L_1, L_2, L_3, L_4)$ with L_1, L_2, L_3, L_4 the Lipschitz constants of $-\gamma$log det(C^TLC+J), tr$(\tilde{X}^TC^TLC\tilde{X})$,

$\|C\tilde{X} - X\|_F^2$, $\|C^T\|_{1,2}^2$respectively. More details are deferred to the appendix A.4. The majorised function $g(C|C^{(t)})$ satisfies all the required properties listed in (9) (a)-(c). After ignoring the constant term, the majorised problem of (10) is

$$\underset{C \in \mathcal{S}_c}{\text{minimize}} \quad \frac{1}{2} C^T C - C^T A \tag{12}$$

where $A = C^{(t)} - \frac{1}{L}\nabla f(C^{(t)})$ and $\nabla f(C^{(t)}) = -2\gamma L C^{(t)}(C^{(t)T} L C^{(t)} + J)^{-1} + \alpha \left(C^{(t)}\tilde{X} - X\right)\tilde{X}^T + 2LC^{(t)}\tilde{X}\tilde{X}^T + \lambda C^{(t)}\mathbf{1}$ where $\mathbf{1}$ is all ones matrix of dimension $k \times k$. Note that since $C \geq 0$, it implies that $|C_{ij}| = C_{ij}$ hence $\|C^T\|_{1,2}^2$ is differentiable.

Lemma 3. *By using KKT optimality condition we can obtain the optimal solution of* (12) *as*

$$C^{(t+1)} = \left(C^{(t)} - \frac{1}{L}\nabla f\left(C^{(t)}\right)\right)^+ \tag{13}$$

where $(X_{ij})^+ = \max(\frac{X_{ij}}{\|[X^T]_i\|_2}, 0)$ *and* $[X^T]_i$ *is the* i*-th row of matrix* X.

Proof: The proof is deferred to the appendix A.3.

Remark The update rule above will be able to learn a balanced mapping, i.e., the loading matrix C belongs to the set (3). Let us begin by expanding $\|C^T\|_{1,2}^2$ as $\|C^T\|_{1,2}^2 = \sum_{i=1}^p (\sum_{j=1}^k C_{ij})^2 = \|C\|_F^2 + \sum_{i \neq j}\langle C_i, C_j\rangle$ for $i, j = 1, 2, ...k$, and analyze the possible solutions under the constraints and the objective of the given problem (10). The trivial solution of all zero $C = \mathbf{0}_{p\times k}$ and any C with zero column vector i.e. $C_i = 0\ \forall i = 1, 2, \ldots p$ will make the term $(C^T L C + J)$ rank deficient and the $\log\det(\cdot)$ term become infeasible and thus ruled out. Moreover, the minimization of $\|C\tilde{X} - X\|_F^2 = \sum_{i=1}^p ([C^T]_i\tilde{X} - X_i)^2$ ensures that no row of C matrix will be zero. Also, as $C \geq 0$, $C \neq \mathbf{0}_{p\times k}$, and from the property of Frobenius norm it implies that $\|C\|_F^2 \neq 0$, thus the only possibility to minimize (10) is to get $C \geq 0$ such that $\sum_{i\neq j}\langle C_i, C_j\rangle = 0$. This implies that columns of the loading matrix C are orthogonal to each other. Finally, the orthogonality of columns combined with $C \geq 0$ implies that in each row there is only one non-zero entry. Thus, the learned C matrix belongs to the set (3), and a more detailed discussion is provided in (Kumar et al., 2023).

3.6. Update of $\tilde{X}$

Fixing C, we obtain the following problem for $\tilde{X}$:

$$\min_{\tilde{X}} f(\tilde{X}) = \text{tr}(\tilde{X}^T C^T L C \tilde{X}) + \frac{\alpha}{2}\|C\tilde{X} - X\|_F^2 \tag{14}$$

As $C^T L C$ and $C^T C$ are the positive semi-definite and definite matrices, respectively, the Hessian of $f(\tilde{X})$ i.e., $\nabla^2 f(\tilde{X}) = 2C^T L C + \alpha C^T C$ is a positive definite matrix. That implies that (14) is a strongly convex optimization problem. For this, we can easily get the closed-form solution by setting the gradient to zero, i.e., $2C^T L C\tilde{X} + \alpha C^T(C\tilde{X} - X) = 0$, we get

$$\tilde{X}^{t+1} = \left(\frac{2}{\alpha} C^T L C + C^T C\right)^{-1} C^T X \tag{15}$$

Algorithm 1 FGC Algorithm

Input: $\mathcal{G}(X, L), \alpha, \gamma, \lambda$

$t \leftarrow 0$;

while *stopping criteria not met* **do**

  Update C^{t+1} and $\tilde{X}^{t+1}$ as in (13) and (15) respectively.

  $t \leftarrow t + 1$;

end

Output: C, L_c, and $\tilde{X}$

Algorithm 1 summarizes the implementation of the featured graph coarsening (FGC) method. The worst-case computational complexity is $\mathcal{O}(p^2 k)$ which is due to the matrix multiplication in the gradient of $f(C)$.

Theorem 1. *The sequence* $\{C^{(t)}, \tilde{X}^{(t)}\}$ *generated by Algorithm 1 converges to the set of Karush–Kuhn–Tucker (KKT) points of Problem* (7).

Proof: The proof is deferred to the appendix A.6.

3.6.1. Similarity Guarantee

Lemma 4. *Consider* $\tilde{X}^*$ *be the minimizer of problem* (14)*,* L *and* L_c *be the graph matrix of original and coarsened graph respectively,* X *is the feature matrix of the original graph then the two norms* $\|\tilde{X}^*\|_{L_c}$ *and* $\|X\|_L$ *with respect to* L_c *and* L *satisfy the following inequality:*

$$\|X\|_L \geq \|\tilde{X}^*\|_{L_c} \tag{16}$$

where $\|X\|_L^2 = \text{tr}(X^T L X)$ and $\|\tilde{X}^*\|_{L_c}^2 = \text{tr}(\tilde{X}^{*T} L_c \tilde{X}^*)$
Proof: The proof is deferred to the appendix A.7.

Theorem 3.1. ***ϵ-similarity*** *The coarsened graph data* $\mathcal{G}_c(L_c, \tilde{X})$ *is* ϵ *similar to the original graph data* $\mathcal{G}(L, X)$*, i.e., there exist an* $0 \leq \epsilon < 1$ *such that*

$$(1 - \epsilon)\|X\|_L \leq \|\tilde{X}\|_{L_c} \leq (1 + \epsilon)\|X\|_L$$

Proof: The proof is deferred to the appendix A.8.

4. Experiments

In this section, we demonstrate the effectiveness of the proposed algorithm through a comprehensive set of experiments conducted on both real and synthetic graph data sets. We compare the proposed Featured Graph Coarsening (FGC) method by benchmarking against the state-of-the-art

methods, Local Variation Edges (LVE), Local Variation Neighbourhood (LVN), proposed in (Loukas, 2019), and GOREN (Cai et al., 2021). Throughout all the experiments, the FGC has shown superior performance.

Datasets: Real and synthetic datasets, dataset(p, m, n) where p is the number of nodes, m is the number of edges and n is the number of features, used in our experiments are: (i) Cora (2708,5278,1433), (ii) Citeseer (3312,4536,3703), (iii) Polblogs (1490,16715,5000), (iv) ACM (3025,13128,1870), (v) Erdos Renyi (ER) $(1000, 25010, 5000)$, (vi) Watts Strogatz (WS) $(1000, 5000, 5000)$, (vii) Barabasi Albert (BA) $(1000, 9800, 5000)$, (viii) Random Geometric Graph (RGG) $(1000, 7265, 5000)$, (ix) Minnesota $(2642, 3304, 5000)$, (x) Yeast $(2361, 13292, 5000)$, (xi) Airfoil $(4253, 12289, 5000)$ and (xii) Bunny $(2503, 78292, 5000)$ (xiii) Pubmed(19717,88648,500) $(34493, 247962, 8, 415)$ (xiv) Coauthor Physics(Co-phy) $(34493, 247962, 8, 415)$. The features of Polblogs, ER, WS, BA, RGG, Yeast, Minnesota, Airfoil and Bunny are generated as following $X \sim \mathcal{N}(\mathbf{0}, L^\dagger)$ (5), where L is the Laplacian matrix of the given graph. Further details of datasets are in the appendix B.

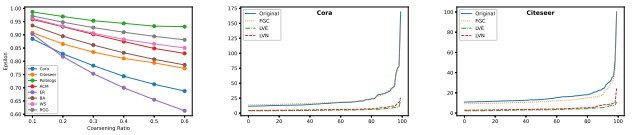
REE, DE, HE and RE analysis: We use relative eigenerror (REE), Reconstruction error (RE), Dirichlet energy (DE) of $\mathcal{G}_c$, and, Hyperbolic error (HE) as the metrics to evaluate the performance of coarsening algorithms as shown in Table 1. Lower values will indicate that the coarsened graph has better preserved the properties of the original graph. The baseline method only uses adjacency matrix information for performing coarsening. Once the coarsening matrix $P = C^\dagger$ is learned, which establishes the linear mapping of the nodes to the supernodes. The matrix P is used further for the coarsening of the feature matrix as $\tilde{X} = PX$.

Comparison with deep learning based methods: We have compared FGC (proposed) against the GOREN(Cai et al., 2021), a deep learning-based graph coarsening approach, on real datasets. Due to the unavailability of their code, we compared only REE because it is the only metric they have computed in their paper among REE, DE, RE, and HE. Their results of REE are taken directly from their paper. It is evident in Table 3 that FGC outperforms GOREN.

REE	Minnesota	Airfoil	Yeast	Bunny
G.(LVN/LVE)	0.38/0.45	0.36/0.33	0.13/0.39	0.16/0.05
FGC	**0.08**	**0.10**	**0.01**	**0.04**

Table 3: REE values of FGC (proposed) and GOREN (Cai et al., 2021) for $r = 0.5$. More results on $r = 0.3, 0.7$ are in the appendix B.1. It is evident that FGC outperforms GOREN.

ϵ-Similarity and Spectral Similarity: Here we evaluate the FGC algorithm for ϵ-similarity as discussed in Theorem (3.1) and spectral similarity. The spectral similarity



(a) $\epsilon-$similarity (b) Eigenvalues plots (c) Eigenvalues plots

Figure 2: Figure (a), the ϵ values lying between $[0, 1)$ indicates that the coarsened graph $\mathcal{G}_c$ learned by FGC (proposed) and $\mathcal{G}$ are similar in terms of Drichlet energy as in Theorem 3.1. The top 100 eigenvalues of L and L_c obtained by FGC, LVN, and LVE are plotted in Figure (b) Cora dataset and Figure (c) Citeseer dataset. This indicates that the FGC better preserves the spectral properties than the existing state-of-the-art methods. Results with more datasets are provided in the appendix B.2.

plots in Figure 2 are obtained for three coarsening methods FGC (proposed), LVE and LVN, and the coarsening ratio is chosen as $r = 0.3$. Experiments on other datasets and coarsening ratios are shown in the appendix B.1.

Clustering: The input is a social network of a university Karate club consisting of 34 nodes, the goal is to classify the nodes into two groups, see Figure 3. For the FGC algorithm feature matrix, X of size $34 \times n$ is generated by sampling from $X \sim \mathcal{N}(\mathbf{0}, L^\dagger)$, where L is the Laplacian matrix of the given network. The result here is shown for $n = 600$, however, an n of the order of $5 * 34$ has been observed to perform well. The FGC result also demonstrates that the features may help in improving the graph-based task, and for some cases like the one presented here the features can also be artificially generated governed by the smoothness and homophily properties.

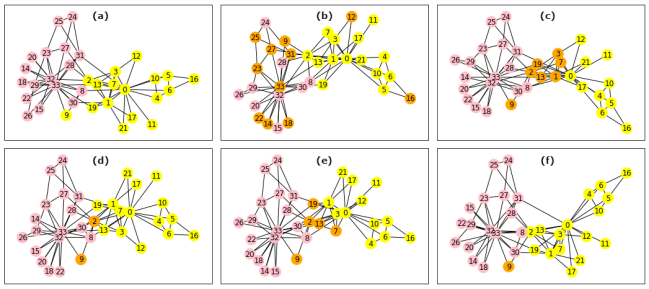


Figure 3: This figure evaluates the clustering performance of the FGC algorithm on the classic Zachary's karate club dataset (Zachary, 1977): (a) Ground truth, (b) Graclus(Dhillon et al., 2007), (c) spectral clustering ratio cut(Ng et al., 2001) (d) spectral clustering normalized cut(Ng et al., 2001) (e) LVN, and (f) FGC (Proposed). Orange nodes indicate misclassified points, FGC demonstrates a better performance, it resulted in 1 misclassified point, while the number of misclassified points for (b), (c), (d) and (e) are 11, 7, 2 and 5, respectively. The clustering experiments on Polblogs dataset are in the appendix B.5.

Dataset	r=$\frac{k}{p}$	REE($L, L_c, 100$)		DE in 10^4		HE		RE in $\log(\cdot)$	
		FGC	LVN/LVE	FGC	LVN/LVE	FGC	LVN/LVE	FGC	LVN/LVE
Cora	0.7	**0.04**	0.33/0.29	**0.75**	10/9.9	**0.72**	1.39/1.42	**1.91**	2.92/2.95
	0.5	**0.051**	0.51/0.53	**0.69**	6.1/5.81	**1.18**	2.29/2.37	**2.78**	3.63/3.67
	0.3	**0.058**	0.65/0.71	**0.66**	3.2/2.8	**1.71**	2.94/3.08	**3.28**	3.77/3.79
Citeseer	0.7	**0.012**	0.32/0.29	**0.71**	13/14	**0.85**	1.68/1.63	**1.32**	2.56/2.51
	0.5	**0.04**	0.54/0.55	**0.69**	7.5/7.1	**1.05**	2.43/2.40	**1.61**	2.87/2.90
	0.3	**0.05**	0.72/0.76	**0.59**	3.1/2.9	**1.80**	3.25/3.41	**2.41**	3.04/3.04
Poblogs	0.7	**0.001**	0.50/0.35	**3.2**	607/656	**1.73**	2.33/2.39	**5.1**	7.27/7.11
	0.5	**0.007**	0.73/0.67	**3.0**	506/468	**2.70**	2.73/2.58	**6.2**	7.42/7.42
	0.3	**0.01**	0.86/0.96	**2.6**	302/115	**2.89**	3.07/3.69	**6.3**	7.50/7.51
ACM	0.7	**0.002**	0.38/0.14	**1.7**	72/93.4	**0.45**	2.13/1.63	**2.42**	5.05/4.66
	0.5	**0.034**	0.66/0.42	**1.5**	30/43	**0.98**	3.10/2.55	**3.78**	5.35/5.18
	0.3	**0.036**	0.92/0.88	**0.5**	5.7/7.5	**1.86**	4.867/4.43	**4.77**	5.44/5.42

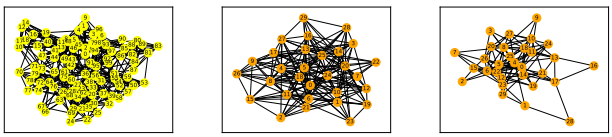
Table 1: This table summarizes the REE, DE, HE and RE values obtained by FGC (proposed), LVN and LVE on different coarsening ratios (r) for real graph datasets. It is evident that FGC (proposed) outperforms state-of-the-art methods significantly. Experiments on other benchmark algorithms are in the appendix B.1.

Dataset	r	GCN	GAT	APPNP
Cora	0.3	85.79±0.2	85.62±0.2	87.25±0.3
Citeseer	0.3	74.64±1.3	74.81±0.8	74.36±0.4
Co-Phy	0.3	94.27±0.2	94.98±0.6	94.23±0.3
Pubmed	0.3	80.73±0.4	81.02±0.7	80.65±0.5

Table 2: The table summarizes the node classification performance of the FGC with different GNN architectures. It is evident that the proposed FGC algorithm is compatible with different GNN architectures and shows similar performance.

4.1. Visualization of sparse coarsened graph

Given a sparse original graph, we can learn a sparse coarsened graph by adding the following regularizer $h(L_c) = \|C^T L C\|_F^2$ in the objective function (7). The regularizer is differentiable and convex in C. The update rule of loading matrix C is modified by adding $2LC^{(t)}(C^{(t)T}LC^{(t)})$ in $\nabla f(C^{(t)})$, while update rule of $\tilde{X}$ remains the same. Next, a visualization for 100 nodes stochastic block model graph coarsened into a 30 nodes graph using i) FGC without sparsity and ii) FGC with sparsity regularizer (FGCS) as $h(L_c) = ||C^T L C||_F^2$ is provided in the Figure 4.



(a) Original graph $\mathcal{G}$ (b) $\mathcal{G}_c$ without sparsity (c) $\mathcal{G}_c$ with sparsity

Figure 4: This figure shows the visualization of 100 nodes stochastic block model graph and coarsened graphs obtained using FGC with and without sparsity regularizer for a coarsening ratio of 0.3.

4.2. Node classification using FGC

In this section, we perform node classification using GNN, in which the goal is to predict a label for unlabelled nodes of a graph. The GNN is trained using the coarsened graph, and the trained model is used to assign labels to the unlabelled nodes of the original graph. The steps to perform node classification are (i) we learn a coarsen graph $G_c = (\tilde{V}, \tilde{E}, \tilde{W}, \tilde{X})$ using FGC algorithm with sparsity regularizer having super-node labels $\tilde{Y}$. The labels for super-nodes can be selected using $\tilde{Y} = \text{argmax}(PY)$ (Huang et al., 2021) where P is the coarsening matrix. (ii) Graph neural network (GNN) is trained using coarsened graph data $\mathcal{G}_c$. (iii) We assign labels to each node of the original graph using the trained GNN model. The learning rate and decay rate used in the node classification experiments are 0.01 and 0.0001, respectively. All the results are calculated using 10-fold cross-validation. It is evident in table Table 4 that FGC outperforms state-of-the-art graph coarsening algorithms.

Next, we used different GNN architectures like GCN (Kipf & Welling, 2016), GAT (Velivcković et al., 2017), and APPNP (Gasteiger et al., 2018) to train our GNN and perform the node classification task. Table 2 demonstrates that FGC is compatible with different widely used GNN architectures, giving almost similar Node Classification accuracy across all the datasets.

Algorithm 2 Algorithm for node classification

Input: $G = (V, E)$, features X, labels Y
Apply a coarsening algorithm to learn P; $P = C^{\dagger}$.;
Compute feature matrix of a coarsened graph, $X' = PX$;
Compute labels of coarsened graph, $Y' = \arg\ max(PY)$
Learn W^* matrix to minimize $\ell(GCN_{Gc}(W^*), Y')$
Output: Trained weight matrix W^*

Dataset	r	GCOND	SCAL	FGC
Cora	0.3	81.56± 0.6	79.42±1.71	**85.79±0.24**
	0.1	81.37±0.40	71.38±3.62	**81.46±0.79**
	0.05	79.93±0.44	55.32±7.03	**80.01±0.51**
Citeseer	0.3	72.43±0.94	68.87±1.37	**74.64±1.37**
	0.1	70.46±0.47	71.38±3.62	**73.36±0.53**
	0.05	64.03±2.4	55.32±7.03	**71.02±0.96**
Co-phy	0.05	93.05±0.26	73.09±7.41	**94.27±0.25**
	0.03	92.81±0.31	63.65±9.65	**94.02±0.20**
	0.01	92.79±0.4	31.08±2.65	**93.08±0.22**
Pubmed	0.05	78.16±0.30	72.82±2.62	**80.73±0.44**
	0.03	78.04±0.47	70.24±2.63	**79.91±0.30**
	0.01	77.2±0.20	54.49±10.5	**78.42±0.43**
Co-CS	0.05	86.29±0.63	34.45±10.0	**89.60±0.39**
	0.03	86.32±0.45	26.06±9.29	**88.29±0.79**
	0.01	84.01±0.02	14.42±8.5	**86.37±1.36**

Table 4: The table summarizes the node classification accuracy on both small and large datasets for the FGC with sparsity regularizer, GCOND (Jin et al., 2021), SCAL (Huang et al., 2021). It is evident that the FGC outperforms state-of-the-art methods. However, we have compared only with GCOND and SCAL as these are the most recent technique for graph node classification using the coarsened graph.

4.3. Time Complexity

Consider the input graph with p nodes, E_1 edges, and a feature vector of size n for each node. If the GCN has l number of layers, the total time complexity to perform node classification using GCN is $\mathcal{O}(lp^2n + lpE_1n)$. However, the time complexity to perform coarsening as well as node classification is $\mathcal{O}(p^2k + lk^2n + lkE_2n)$ where k is the number of nodes of the coarsened graph and E_2 is the number of edges of the coarsened graph. Since $(p >> k)$ and $E_1 >> E_2$ and if choose k such that $k < n$, then the time complexity to perform coarsening and node classification is very less as compared to performing node classification using the original graph. It is evident in Table 5 that the proposed FGC algorithm is much faster than baseline methods.

Dataset(τ)	GCOND	SCAL	FGC	GCN(wh.data)
Cora	329.8	27.7	**1.66**	2.86
Citeseer	331.3	56.2	**2.22**	5.24
Pubmed	202.0	54.0	**30.4**	58.8
Co-CS	1600	180	**34.4**	72.3

Table 5: The table summarizes the time complexity comparison between proposed FGC and baseline algorithms for a coarsening ratio of $r = 0.05$, where τ(in sec.) is the time required to perform coarsening and classification, and wh.data represents whole dataset. It is evident that the proposed FGC is much faster than the existing baselines.

Dataset	r	Prop.FGC	GCN(Whole dataset)
Cora	0.3	85.79± 0.24	89.50 ± 1.23
Citeseer	0.3	74.64± 1.37	78.08 ± 1.96
Pubmed	0.05	80.73 ± 0.44	88.89 ± 0.59
Co-phy	0.05	94.27± 0.25	96.22 ± 0.72

Table 6: The table summarizes the node classification performance comparison of the proposed FGC with the original Graph (No Coarsening). It is evident that the node classification accuracy obtained using the coarsened graph with the proposed FGC approach is very close to the one obtained with the full graph that is the original graph.

5. Conclusion

We introduced a novel framework for coarsening graph data named Featured Graph Coarsening (FGC) which considers both the graph matrix and feature matrix jointly. The featured graph coarsening is formulated as a multi-block non-convex optimization problem for which we developed an efficient algorithm by bringing in techniques from alternate minimization, majorization-minimization, and $\log$ determinant frameworks. The developed algorithm is provably convergent and ensures the necessary properties in the coarsened graph data. Also, the FGC method is the first coarsening method that guarantees that the original graph and coarsened graph are ϵ-similar for $\epsilon \in [0, 1)$. Extensive experiments with both real and synthetic datasets demonstrate the superiority of the proposed FGC framework over existing state-of-the-art methods. The proposed approach for graph coarsening will be of significant interest to the graph machine-learning community.

6. Acknowledgments

We would like to thank the reviewers for their suggestions to improve this paper. We would also like to thank to whole MISN lab members for their contributions to the discussions regarding the paper and code implementation. This work is supported by the DST Inspire faculty grant MI02322G.

References

Beck, A. and Pan, D. *Convergence of an Inexact Majorization-Minimization Method for Solving a Class of Composite Optimization Problems*, pp. 375–410. 01 2018. ISBN 978-3-319-97477-4. doi: 10.1007/978-3-319-97478-1_13.

Bravo Hermsdorff, G. and Gunderson, L. A unifying framework for spectrum-preserving graph sparsification and coarsening. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.),

Advances in Neural Information Processing Systems, volume 32. Curran Associates, Inc., 2019.

Briggs, W. L., Henson, V. E., and McCormick, S. F. *A multigrid tutorial*. Society for Industrial and Applied Mathematics., 2000.

Cai, C., Wang, D., and Wang, Y. Graph coarsening with neural networks. In *International Conference on Learning Representations*, 2021.

Chen, J., Saad, Y., and Zhang, Z. Graph coarsening: from scientific computing to machine learning. *Journal of the Spanish Society of Applied Mathematics (SeMA)*, 79(1): 187–223, 2022.

Chung, F. R. and Graham, F. C. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Dhillon, I. S., Guan, Y., and Kulis, B. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11):1944–1957, 2007.

Dong, X., Thanou, D., Frossard, P., and Vandergheynst, P. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

Fortunato, S. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Gavish, M., Nadler, B., and Coifman, R. R. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *International Conference on Machine Learning*, 2010.

Hackbusch, W. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

Hajiabadi, M., Singh, J., Srinivasan, V., and Thomo, A. Graph summarization with controlled utility loss. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 536–546, 2021.

Hendrickson, B., Leland, R. W., et al. A multi-level algorithm for partitioning graphs. *SC*, 95(28):1–14, 1995.

Huang, Z., Zhang, S., Xi, C., Liu, T., and Zhou, M. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 675–684, 2021.

Jin, W., Zhao, L., Zhang, S., Liu, Y., Tang, J., and Shah, N. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580*, 2021.

Jin, W., Tang, X., Jiang, H., Li, Z., Zhang, D., Tang, J., and Yin, B. Condensing graphs via one-step gradient matching. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 720–730, 2022.

Kalofolias, V. How to learn a graph from smooth signals. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 920–929, Cadiz, Spain, 09–11 May 2016. Proceedings of Machine Learning Research.

Kang, S., Lee, K., and Shin, K. Personalized graph summarization: formulation, scalable algorithms, and applications. *arXiv preprint arXiv:2203.14755*, 2022.

Karypis, G. and Kumar, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Ko, J., Kook, Y., and Shin, K. Incremental lossless graph summarization. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 317–327, 2020.

Kumar, M., Sharma, A., and Kumar, S. A unified framework for optimization-based graph coarsening. *Journal of Machine Learning Research*, 24(118):1–50, 2023.

Kumar, S., Ying, J., de Miranda Cardoso, J. V., and Palomar, D. Structured graph learning via laplacian spectral constraints. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Kumar, S., Ying, J., de Miranda Cardoso, J. V., and Palomar, D. P. A unified framework for structured graph learning via spectral constraints. *Journal of Machine Learning Research*, 21(22):1–60, 2020.

Kushnir, D., Galun, M., and Brandt, A. Fast multiscale clustering and manifold identification. *Pattern Recognition*, 39(10):1876–1891, 2006.

Lafon, S. and Lee, A. B. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE transactions on pattern analysis and machine intelligence*, 28(9):1393–1403, 2006.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Liu, Y., Safavi, T., Dighe, A., and Koutra, D. Graph summarization methods and applications: A survey. *ACM computing surveys (CSUR)*, 51(3):1–34, 2018.

Loukas, A. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Loukas, A. and Vandergheynst, P. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pp. 3237–3246. Proceedings of Machine Learning Research, 2018.

Ma, T. and Chen, J. Unsupervised learning of graph hierarchical abstractions with differentiable coarsening and optimal transport, 2020.

Ma, T. and Chen, J. Unsupervised learning of graph hierarchical abstractions with differentiable coarsening and optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8856–8864, 2021.

Ming, D., Ding, C., and Nie, F. A probabilistic derivation of lasso and l12-norm feature selections. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4586–4593, 2019.

Ng, A., Jordan, M., and Weiss, Y. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

Purohit, M., Prakash, B. A., Kang, C., Zhang, Y., and Subrahmanian, V. Fast influence-based coarsening for large networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1296–1305, 2014.

Qiu, Y., Zhou, G., and Xie, K. Deep approximately orthogonal nonnegative matrix factorization for clustering. *arXiv preprint arXiv:1711.07437*, 2017.

Rajawat, K. and Kumar, S. Stochastic multidimensional scaling. *IEEE Transactions on Signal and Information Processing over Networks*, 3(2):360–375, 2017.

Razaviyayn, M., Hong, M., and Luo, Z.-Q. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23, 09 2012. doi: 10.1137/120891009.

Riondato, M., García-Soriano, D., and Bonchi, F. Graph summarization with quality guarantees. *Data mining and knowledge discovery*, 31(2):314–349, 2017.

Ruge, J. W. and Stüben, K. Algebraic multigrid. In *Multigrid methods*, pp. 73–130. SIAM, 1987.

Shuman, D. I., Faraji, M. J., and Vandergheynst, P. A multiscale pyramid transform for graph signals. *IEEE Transactions on Signal Processing*, 64(8):2119–2134, 2015.

Sun, Y., Babu, P., and Palomar, D. P. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2017. doi: 10.1109/TSP.2016.2601299.

Velivcković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. Heterogeneous graph attention network. In *The world wide web conference*, pp. 2022–2032, 2019.

Wang, X., Pun, Y.-M., and So, A. Learning graphs from smooth signals under moment uncertainty. 05 2021.

Ying, J., de Miranda Cardoso, J. V., and Palomar, D. Nonconvex sparse graph learning under laplacian constrained graphical model. *Advances in Neural Information Processing Systems*, 33:7101–7113, 2020.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68 (1):49–67, 2006.

Zachary, W. W. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, 33(4):452–473, 1977.

Zhu, P., Zhu, W., Hu, Q., Zhang, C., and Zuo, W. Subspace clustering guided unsupervised feature selection. *Pattern Recognition*, 66:364–374, 2017.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.

A. Appendix

A.1. Proof of Lemma 1

The matrix $L \in \mathbb{R}^{p\times p}$ is the Laplacian matrix of a connected graph having p nodes. From (1) it is implied that $L = L^T$, L is positive semi-definite matrix with rank$(L) = p-1$ and $L \cdot t\mathbf{1}_p = \mathbf{0}_p$, where $t \in \mathbf{R}$ and $\mathbf{1}_p$ and $\mathbf{0}_p$ are the all one and zero vectors of size p. In addition, we also have $L \cdot \mathbf{u}_p \neq \mathbf{0}_p$ for $\mathbf{u}_p \neq t\mathbf{1}_p$, this means that there is only one zero eigenvalues possible and the corresponding eigenvector is a constant vector. In order to establish L_c is the connected graph Laplacian matrix of size k, we need to prove that $L_c \in$ (1) and rank$(L_c) = k-1$.

We begin by using the Cholesky decomposition of the Laplacian matrix L, as $L = S^T S$. Next, we can write $C^T LC$ as

$$L_c = C^T LC = C^T S^T SC \tag{17}$$
$$= Z^T Z \tag{18}$$

where $Z = SC$ and $C^T LC = Z^T Z$ imply that L_c is a symmetric positive semidefinite matrix. Now, using the property of C, i.e, $C \cdot t\mathbf{1}_k = t\mathbf{1}_p$ as in (3). In each row of the loading matrix C, there is only one non zero entry and that entry is 1 which implies that $C \cdot \mathbf{1}_k = \mathbf{1}_p$ and $C^T LC \cdot \mathbf{1}_k = C^T L \cdot \mathbf{1}_p = \mathbf{0}_k$ which imply that the row sum of $C^T LC$ is zero and constant vector is the eigenvector corresponding to the zero eigenvalue. Thus L_c is the Laplacian matrix.

Next, we need to prove that L_c is a connected graph Laplacian matrix for that we need to prove that rank$(L_c) = k-1$. Note that, $C \cdot \mathbf{u}_k = \mathbf{u}_p$ if and only if $\mathbf{u}_k = t\mathbf{1}_k$ and $\mathbf{u}_p = t\mathbf{1}_p$ and $C \cdot \mathbf{u}_k \neq \mathbf{u}_p \; \forall \; \mathbf{u}_k \neq t\mathbf{1}_k$ and $\mathbf{u}_p \neq t\mathbf{1}_p$ where $t \in \mathbb{R}$ which implies that $C \cdot \mathbf{u}_k = \mathbf{u}_p$ holds only for a constant vector $\mathbf{u}_k$. And thus, $C^T LC \cdot \mathbf{u}_k = \mathbf{0}_k$, for constant vector $\mathbf{u}_k$. This concludes that the constant vector is the only eigenvector spanning the nullspace of L_c which concludes that the rank of $C^T LC$ is $k-1$ which completes the proof.

A.2. Proof of Lemma 2

We prove the convexity of $-\gamma$log det$(C^T LC + J)$ using restricting function to line i.e. A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $g : \mathbb{R} \to \mathbb{R}$ is convex where,

$$g(t) = f(z + tv), \{z \in dom(f), t \in dom(g), v \in \mathbb{R}^n\} \tag{19}$$

Since L is the Laplacian of connected original Graph $\mathcal{G}$ and Laplacian of coarsened graph $\mathcal{G}_c$ is $L_c = C^T LC$ which also represents a connected graph and proof is given in the appendix A.1. Using the property of the connected graph Laplacian matrix, L_c is a symmetric positive semi-definite matrix and has a rank $k-1$. Adding $J = \frac{1}{k}\mathbf{1}_{k\times k}$ which is a rank 1 matrix in L_c increases rank by 1. $L_c + J$ becomes symmetric and positive definite matrix and we can rewrite $L_c + J = C^T LC + J = Y^T Y$. Now, we can rewrite $-\gamma$log det$(C^T LC + J)$ as

$$f(Y) = -\gamma\text{log det}(C^T LC + J) = -\gamma\text{log det}(Y^T Y) \tag{20}$$

Consider Y=Z+tV and put it in (20). However, Z and V are constant so function in Y becomes function in t i.e. $g(t)$ is

$$g(t) = -\gamma\text{log det}((Z + tV)^T (Z + tV)) \tag{21}$$
$$= -\gamma\text{log det}(Z^T Z + t(Z^T V + V^T Z) + t^2 V^T V) \tag{22}$$
$$= -\gamma\text{log det}(Z^T (I + t(VZ^{-1} + (VZ^{-1})^T) + t^2 (Z^{-1})^T V^T V Z^{-1})Z) \tag{23}$$
$$= -\gamma(\text{log det}(Z^T Z) + \text{log det}(I + t(P + P^T) + t^2 P^T P)) \tag{24}$$

$$g(t) = -\gamma(\log\det(Z^TZ) + \log\det(QQ^T + 2tQ\Lambda Q^T + t^2Q\Lambda^2Q^T)) \tag{25}$$
$$= -\gamma(\log\det(Z^TZ) + \log\det(Q(I + 2t\Lambda + t^2\Lambda^2)Q^T)) \tag{26}$$
$$= -\gamma\log\det(Z^TZ) - \gamma\sum_{i=1}^{n}\log(1 + 2t\lambda_i + t^2\lambda_i^2) \tag{27}$$

On putting $VZ^{-1} = P$ in (23) to get (24). Using eigenvalue decomposition of P matrix i.e. $P = Q\Lambda Q^T$ and $QQ^T = I$ and putting the values of P and I in (24) to get (25). The second derivative of g(t) with respect to t is

$$g^{''}(t) = \sum_{i=1}^{n}\frac{2\lambda_i^2(1 + t\lambda_i)^2}{(1 + 2t\lambda_i + t^2\lambda_i^2)^2} \tag{28}$$

It is clearly seen that $g^{''}(t) \geq 0, \forall t \in \mathbb{R}$ so it is a convex function in t. Now, using the restricting function to line property if $g(t)$ is convex in t then $f(Y)$ is convex in Y. Consider $Y = L^{\frac{1}{2}}C + \frac{1}{\sqrt{kp}}\mathbf{1}_{P\times k}$ so,

$$Y^TY = (L^{\frac{1}{2}}C + \frac{1}{\sqrt{kp}}\mathbf{1}_{P\times k})^T(L^{\frac{1}{2}}C + \frac{1}{\sqrt{kp}}\mathbf{1}_{P\times k}) \tag{29}$$
$$= C^TLC + \frac{1}{kp}(p\mathbf{1}_{k\times k}) + \frac{1}{\sqrt{kp}}\mathbf{1}_{P\times k}^TL^{\frac{1}{2}}C + \frac{1}{\sqrt{kp}}C^T(L^{\frac{1}{2}})^T\mathbf{1}_{P\times k} \tag{30}$$
$$= C^TLC + \frac{1}{k}\mathbf{1}_{k\times k} \tag{31}$$
$$= C^TLC + J \tag{32}$$

L is a Laplacian matrix so $L^{\frac{1}{2}}$ is also Laplacian matrix and using the property of Laplacian matrix i.e. $L^{\frac{1}{2}}.\mathbf{1}_{p\times k} = \mathbf{0}_{p\times k}$ and $\mathbf{1}_{p\times k}^T.L^{\frac{1}{2}} = \mathbf{0}_{k\times p}$ in (30), we get (31). Since $Y = L^{\frac{1}{2}}C + \frac{1}{\sqrt{pk}}\mathbf{1}_{p\times k}$ and $f(Y)$ is convex in Y and C is a affine transformation of Y so $-\gamma\log\det(C^TLC + J)$ is a convex function in C.

A.3. Proof of Lemma 3

The Lagrangian function of (12) is:

$$L(C, \tilde{X}, \boldsymbol{\mu}_1) = \frac{1}{2}C^TC - C^TA - \boldsymbol{\mu}_1^\top C + \boldsymbol{\mu}_2^T\Big[\|C_1^T\|_2^2 - 1, \dots \|C_p^T\|_2^2 - 1\Big]^T \tag{33}$$

where $\boldsymbol{\mu}_1$ is the dual variable. The KKT conditions of (12) is

$$C - A - \boldsymbol{\mu}_1 + 2\Big[\mu_{21}C_1^T, \dots \mu_{2p}C_p^T]^T = 0, \tag{34}$$
$$\boldsymbol{\mu}_2^T\Big[\|C_1^T\|_2^2 - 1 \quad \|C_2^T\|_2^2 - 1 \dots \|C_p^T\|_2^2\Big]^T - 1 = 0, \tag{35}$$
$$\boldsymbol{\mu}_1^\top C = 0, \tag{36}$$
$$C \geq 0, \tag{37}$$
$$\boldsymbol{\mu}_1 \geq 0 \tag{38}$$
$$\|[C^T]_i\|_2^2 \leq 1 \tag{39}$$
$$\boldsymbol{\mu}_2 \geq 0 \tag{40}$$

The optimal solution of C that satisfies all KKT conditions (34)-(40) is

$$C^{t+1} = \frac{(A)^+}{\|[A^T]_i\|_2} \tag{41}$$

where $A = \left(C^{(t)} - \frac{1}{L}\nabla f\left(C^{(t)}\right)\right)^+$ and $\|[A^T]_i\|$ is the i-th row of matrix A. This concludes the proof.

A.4. Lipschitz Continuity of Function $f(C)$ in (11)

Consider the function $-\gamma \log\det(C^T LC + J)$. The Lipschitz constant L_1 of the function $-\gamma \log\det(C^T LC + J)$ is related to the smallest non zero eigenvalue of coarsened Laplacian matrix $C^T LC = L_c$, which is bounded away from $\frac{\delta}{(k-1)^2}$ (Rajawat & Kumar, 2017), where δ is the minimum non zero weight of coarsened graph. However, for practical purposes, the edges with very small weights can be ignored and set to be zero, and we can assume that the non-zero weights of the coarsened graph $\mathcal{G}_c$ are bounded by some constant $\delta \geq 0$. On the other hand, we do not need a tight Lipschitz constant L_1. In fact, any $L_1^{'} \geq L_1$ makes the function $g(C|C^{(t)})$ satisfy (9)(a)-(c).

Now, consider the tr$(\cdot)$ term:

$$\begin{aligned}
\Big|\text{tr}(\tilde{X}^T C1^T LC1\tilde{X}) - \text{tr}(\tilde{X}^T C2^T LC2\tilde{X})\Big| &= \Big|\text{tr}(\tilde{X}^T C1^T LC1\tilde{X}) - \text{tr}(\tilde{X}^T C2^T LC1\tilde{X}) \\
&\quad + \text{tr}(\tilde{X}^T C2^T LC1\tilde{X}) - \text{tr}(\tilde{X}^T C2^T LC2\tilde{X})\Big| \qquad (42)\\
&\leq \Big|\text{tr}(\tilde{X}^T (C1 - C2)^T LC1\tilde{X})\Big| + \Big|\text{tr}(\tilde{X}^T C2^T L(C1 - C2)\tilde{X}\Big| \qquad (43)\\
&\leq \|\text{tr}\|\|\tilde{X}^T (C1 - C2)^T LC1\tilde{X}\|_F + \|\text{tr}\|\|\tilde{X}^T C2^T L(C1 - C2)\tilde{X}\|_F \qquad (44)\\
&\leq ||\text{tr}||||\tilde{X}||_F^2 ||L||||C1 - C2||_F (||C1||_F + ||C2||_F) \qquad (45)\\
&\leq L_2 ||C1 - C2||_F \qquad (46)
\end{aligned}$$

We applied the triangle inequality after adding and subtracting $\text{tr}(\tilde{X}^T C2^T LC1\tilde{X})$ in (42) to get (43). Using the property of the norm of the trace operator i.e. $\|\text{tr}\| = \underset{A\neq 0}{sup} \frac{|\text{tr(A)}|}{||A||_F}$ from $\mathbb{R}^{n\times n}$ to $\mathbb{R}$ in (43) to get (44). Applying the Frobenius norm property i.e. $\|AB\|_F \leq \|A\|_F \|B\|_F$ in (44) to get (45). Since, in each row of C is having only one non zero entry i.e. 1 and rest entries are zero so, $\|C1\|_F = \|C2\|_F = \sqrt{p}$ and putting this in (45), we get (46) where, $L_2 = 2\sqrt{p}\|\text{tr}\|\|\tilde{X}\|_F^2 ||L||_F$.

Next, consider the function $\frac{\alpha}{2}\|C\tilde{X} - X\|_F^2$:

$$\begin{aligned}
\frac{\alpha}{2}\|C\tilde{X} - X\|_F^2 &= \frac{\alpha}{2}\text{tr}((C\tilde{X} - X)^T (C\tilde{X} - X)) \qquad (47)\\
&= \frac{\alpha}{2}\text{tr}(\tilde{X}^T C^T C\tilde{X} - X^T C\tilde{X} + X^T X - \tilde{X}^T C^T X) \qquad (48)\\
&= \frac{\alpha}{2}(\text{tr}(\tilde{X}^T C^T C\tilde{X}) - \text{tr}(\tilde{X}^T C^T X) - \text{tr}(X^T C\tilde{X}) + \text{tr}(X^T X)) \qquad (49)
\end{aligned}$$

With respect to C, $\text{tr}(X^T X)$ is a constant and $\text{tr}(\tilde{X}^T C^T C\tilde{X}), \text{tr}(\tilde{X}^T C^T X), \text{tr}(X^T C\tilde{X})$ are Lipschitz continous function and proof is very similar to the proof of tr$(\cdot)$ as in (42)-(46), and sum of Lipschitz continuous function is Lipschitz continuous so $\frac{\alpha}{2}\|C\tilde{X} - X\|_F^2$ is L_3 Lipschitz continuous.

Finally, consider the function $\frac{\lambda}{2}\|C^T\|_{1,2}^2$. Note that we have $C \geq 0$ means all the elements of C are non-negative, $|C|_{ij} = C_{ij} \geq 0$. With this the ℓ_1-norm becomes summation, and we obtain the following:

$$\begin{aligned}
\|C^T\|_{1,2}^2 &= \sum_{i=1}^{p} (\sum_{j=1}^{k} C_{ij})^2 \qquad (50)\\
&= \sum_{i=1}^{p} ([C^T]_i \mathbf{1})^2 \qquad (51)\\
&= \|C\mathbf{1}\|_F^2 \qquad (52)\\
&= \text{tr}(\mathbf{1}^T C^T C\mathbf{1}) \qquad (53)
\end{aligned}$$

where $\mathbf{1}$ is a vector having all entry 1, $[C^T]_i$ is i-th row of loading matrix C and since each entry of C is $C_{ij} \geq 0$. $\text{tr}(\mathbf{1}^T C^T C\mathbf{1})$ is Lipschitz continuous function and proof is similar to proof of tr$(\cdot)$ as in (42)-(46) so $\|C^T\|_{1,2}^2$ is L-4 Lipschitz continuous function.

Addition of Lipschitz continuous functions is Lipschitz continuous so we can say that $f(C)$ in (10) is L- Lipschitz continuous function where $L = \max(L_1, L_2, L_3, L_4)$.

A.5. $f(C)$ in (10) is strictly convex

log det$(\cdot)$ and trace$(\cdot)$ are convex functions and proof are in Lemma 1 and 2 respectively, also Frobenius and $\ell_{1,2}^2$ norm are convex functions. Consider the term $\|C^T\|_{1,2}^2 = \sum_{i=1}^{p} \|[C^T]_i\|_1^2 > 0$ which implies that $f(C)$ in (10) is strictly convex function.

A.6. Proof of Theorem 1

The Lagrangian function of (7) is

$$L(C, \tilde{X}, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2) = -\gamma \log\det(C^T L C + J) + \frac{\alpha}{2}||X - C\tilde{X}||_F^2 + \text{tr}(\tilde{X}^T C^T L C \tilde{X}) + \frac{\lambda}{2}\sum_{i=1}^{p}\|[C^T]_i\|_1^2 \tag{54}$$
$$-\text{tr}(\boldsymbol{\mu}_1^\top C) + \boldsymbol{\mu}_2^T\Big[(\|C_1^T\|_2^2 - 1), \ldots, (\|C_p^T\|_2^2 - 1)\Big]^T$$

where $\boldsymbol{\mu}_{1(p\times k)}$ and $\boldsymbol{\mu}_{2(p\times 1)}$ are the dual variables.
(1) The KKT conditions with respect to C is

(i) $-2\gamma LC(C^T LC + J)^{-1} + \alpha\left(C\tilde{X} - X\right)\tilde{X}^T + 2LC\tilde{X}\tilde{X}^T + \lambda C\mathbf{1}_{k\times k} - \boldsymbol{\mu}_1 + 2\Big[\mu_{21}C_1^T, \ldots \mu_{2p}C_p^T\Big]^T = 0,$
(ii) $\left\{\mu_{2i}(\|C_i^T\|_2^2 - 1) = 0\right\}_{i=1}^{p}$, $\text{tr}(\boldsymbol{\mu}_1^\top C) = 0$
(iii) $C \geq 0,\ \|[C^T]_i\|_2^2 \leq 1\ \forall\ i = 1, 2, \ldots, p,$
(iv) $\boldsymbol{\mu}_1 \geq 0, \quad \boldsymbol{\mu}_2 \geq 0$

where $\mathbf{1}_{k\times k}$ is a $k \times k$ matrix whose all entry is one. The variable C is derived by using the KKT condition from (13):

$$C^\infty - C^\infty + \frac{1}{L}\Big(-2\gamma LC^\infty\Big((C^\infty)^T LC^\infty + J\Big)^{-1} + \alpha(C^\infty\tilde{X}^\infty - X)(\tilde{X}^\infty)^T + 2LC^\infty\tilde{X}^\infty(\tilde{X}^\infty)^T + \lambda C^\infty\mathbf{1}_{k\times k}\Big) = 0 \tag{55}$$

For $\boldsymbol{\mu}_1 = 0$ and $\mu_{2i}[C^T]_i^\infty = 0\ \forall\ i = 1, 2, \ldots p$, we observe that C^∞ satisfies the KKT condition. The KKT condition with respect to $\tilde{X}$ is

$$2C^T LC\tilde{X} + \alpha C^T(C\tilde{X} - X) = 0$$

The solution of convex optimization problem (14) is

$$\tilde{X}^\infty = \left(\frac{2}{\alpha}(C^\infty)^T LC^\infty + C^T C\right)^{-1}(C^\infty)^T X^\infty$$

which satisfies the KKT condition. This concludes the proof.

A.7. Proof of Lemma 4

Let $f_1(\tilde{X}) = \text{tr}(\tilde{X}^T C^T LC\tilde{X}) = \|\tilde{X}\|_{L_c}^2$ and $f_2(\tilde{X}) = \frac{\alpha}{2}\|C\tilde{X} - C\|_F^2$. Using the property of convexity, at $t-$th iteration, following inequality holds:

$$f_1(\tilde{X}^{t+1}) + f_2(\tilde{X}^{t+1}) \leq f_1(\tilde{X}^t) + f_2(\tilde{X}^t) \leq f_1(\tilde{X}^0) + f_2(\tilde{X}^0) \tag{56}$$

In algorithm 1, we have initialized $\tilde{X}$ as $\tilde{X} = C^\dagger X$, $f_2(\tilde{X}^0) = 0$ and $f_1(\tilde{X}^0) = \|X\|_L^2$ then following inequality holds

$$f_1(\tilde{X}^{t+1}) + f_2(\tilde{X}^{t+1}) \leq f_1(\tilde{X}^0) \rightarrow f_1(\tilde{X}^{t+1}) \leq f_1(\tilde{X}^0) = \|X\|_L^2 \tag{57}$$

and this will hold for all iterations and we get

$$f_1(\tilde{X}^*) = \|\tilde{X}^*\|_{L_c}^2 \leq f_1(\tilde{X}^0) = \|X\|_L^2 \tag{58}$$

Since $\|\tilde{X}^*\|_{L_c}$ and $\|X\|_L$ are positive quantities so,

$$\|\tilde{X}^*\|_{L_c} \leq \|X\|_L \tag{59}$$

A.8. Proof of Theorem 3.1:

We have $\|X\|_L = \sqrt{\text{tr}(X^T L X)}$ and $\|\tilde{X}\|_{L_c} = \sqrt{\text{tr}(\tilde{X}^T L_c \tilde{X})}$. Taking the absolute difference between $\|X\|_L$ and $\|\tilde{X}\|_{L_c}$ and L is a positive semi-definite matrix using Cholesky's decomposition $L = S^T S$ we get:

$$\left| \|X\|_L - \|\tilde{X}\|_{L_c} \right| = \left| \|SX\|_F - \|SP^\dagger PX\|_F \right| \tag{60}$$

$$\leq \|SX - SP^\dagger PX\|_F \leq \epsilon \|X\|_L \tag{61}$$

Also using Lemma 4, we get the following inequality

$$\frac{\left| \|X\|_L - \|\tilde{X}\|_{L_c} \right|}{\|X\|_L} < 1 \tag{62}$$

The equation (61) and (62) implies that the range of $\epsilon \in [0, 1)$. Next, by applying the property of the modulus function in (61), we obtain the following inequality for all the n samples:

$$(1 - \epsilon)\|X\|_L \leq \|\tilde{X}\|_{L_c} \leq (1 + \epsilon)\|X\|_L \tag{63}$$

where $\epsilon \in [0, 1)$ and this concludes the proof.

B. ADDITIONAL EXPERIMENTS

In the following experiments, we apply the FGC algorithm on both real and synthetic datasets. For synthetic datasets feature matrices are generated as $X \sim \mathcal{N}(\mathbf{0}, L^\dagger)$, where L is the Laplacian matrix of $\mathcal{G}$, and weights in their weight matrices are generated randomly and uniformly from a range of (1,10). These results show that FGC performs outstandingly on both real and synthetic datasets. For simplicity, we have chosen lambda to be $\frac{n}{2}$ and tuned other hyperparameters accordingly.
(1) The additional details of real datasets are as follows:

- Cora. Hyperparameters (γ=500, α=500, λ=716.5) used in FGC algorithms. DE of $\mathcal{G}$ is 160963.
- Citeseer. Hyperparameters (γ=500, α=500, λ=1851.5) used in FGC algorithms. DE of $\mathcal{G}$ is 238074.
- Polblogs. Hyperparameters (γ=500, α=500, λ=2500) used in FGC algorithms. DE of $\mathcal{G}$ is 6113760.
- ACM. Hyperparameters (γ=2000, α=500, λ=935) used in FGC algorithms. DE of $\mathcal{G}$ is 1654444.
- Bunny. This dataset consists of p=2503, m=78292, n=5000. Hyperparameters (γ=450, α=500, λ=2500) used in FGC algorithms. DE of $\mathcal{G}$ is 12512526.
- Airfoil. This dataset consists of p=4253, m=12289, n=5000. Hyperparameters (γ=2000, α=600, λ=2500) used in FGC algorithms. DE of $\mathcal{G}$ is 21269451.
- Yeast. This dataset consists of p=2224, m=4416, n=5000. Hyperparameters (γ=350, α=650, λ=2500) used in FGC algorithms. DE of $\mathcal{G}$ is 10697908.
- Minnesota. This dataset consists of p=2642, m=3304, n=5000. Hyperparameters (γ=500, α=550, λ=2500) used in FGC algorithms. DE of $\mathcal{G}$ is 13207844.

(2) The details of synthetic datasets are as follows:

- Erdos Renyi (ER). It is represented as $\mathcal{G}(n, p)$, where $n = 1000$ is the number of nodes and $p = 0.1$ is probability of edge creation. Hyperparameters (γ=500, α=500, λ=10) used in FGC algorithms. DE of $\mathcal{G}$ is 4995707.
- Barabasi Albert (BA). It is represented as $\mathcal{G}(n, m)$, where $n = 1000$ is the number of nodes and $m = 20$ edges are preferentially linked to existing nodes with a higher degree. Hyperparameters(γ=500, α=500, λ=1000) used in FGC algorithms . DE of $\mathcal{G}$ is 4989862.

- Watts Strogatz (WS). It is represented as $\mathcal{G}(n, k, p)$, where $n = 1000$ is the number of nodes, $k = 20$ is nearest neighbors in ring topology connected to each node, $p = 0.1$ is probability of rewiring edges. Hyperparameters(γ=500, α=500, λ=1000) used in FGC algorithm. DE of $\mathcal{G}$ is 4997509.

- Random Geometric Graph (RGG). It is represented as $\mathcal{G}(n, radius)$, where $n = 1000$ is the number of nodes and $radius = 0.1$ is the distance threshold value for an edge to create. Hyperparameters(γ=500, α=500, λ=1000) used in FGC algorithm. DE of $\mathcal{G}$ is 4989722.

B.1. REE and DE analysis

For synthetic datasets also, to measure the spectral similarity and ϵ-similarity of coarsened graph $\mathcal{G}_c$ and original graph $\mathcal{G}$, we evaluate REE$(L, L_c, 100)$, where L and L_c are Laplacian matrices of $\mathcal{G}$ and $\mathcal{G}_c$ respectively. We also compute DE of $\mathcal{G}_c$ to measure its smoothness.

	$r = k/p$	BA	WS	ER	RGG
REE	0.3	0.623	0.202	0.138	0.234
	0.5	0.211	0.109	0.076	0.249
	0.7	0.084	0.201	0.055	0.252
DE	0.3	212117	35948	293116	30112
	0.5	208352	41889	542051	41215
	0.7	201324	44852	534544	36483

Table 7: REE and DE results for synthetic datasets.

We have compared FGC (proposed) algorithm with Kron reduction (Kron) and heavy edge matching (HEM) using REE and DE. The results are shown in the Table 6.

Dataset	r=$\frac{k}{p}$	REE$(L, L_c, 100)$			DE in 10^4		
		FGC	Kron	HEM	FGC	Kron	HEM
Cora	0.7	**0.04**	0.3815	0.3814	**0.75**	9.1	9.1
	0.5	**0.051**	0.579	0.5804	**0.69**	5.5	5.4
	0.3	**0.058**	0.7406	0.774	**0.66**	2.7	2.4
Citeseer	0.7	**0.0124**	0.3153	0.3153	**0.71**	12.9	12.9
	0.5	**0.04**	0.5411	0.542	**0.69**	7.0	7.0
	0.3	**0.05**	0.778	0.807	**0.59**	2.7	2.5
Poblogs	0.7	**0.001**	0.4256	0.4474	**3.2**	752	761
	0.5	**0.007**	0.67	0.708	**3.0**	513	373
	0.3	**0.01**	0.9653	0.929	**2.6**	132	183
ACM	0.7	**0.002**	0.1568	0.1568	**1.7**	94.5	94.5
	0.5	**0.034**	0.400	0.4172	**1.5**	49	46.1
	0.3	**0.036**	0.858	0.935	**0.5**	8.9	5.4

Table 8: This table summarizes the REE and DE values obtained by FGC (proposed), Kron and HEM on different coarsening ratios (r) for standard real graph datasets. It is evident that FGC (proposed) outperforms state-of-the-art methods significantly.

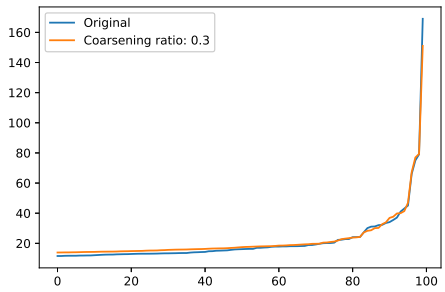
Experiments on Bunny, Airfoil, Yeast and Minnesota comparing FGC (proposed) with GOREN (FGC), GOREN (LVN) and GOREN (LVE) using REE are shown below, where G. stands for GOREN in Table 7.

Dataset	r=$\frac{k}{p}$	REE($L, L_c, 100$)			
		FGC	G.HEM	G.LVN	G.LVE
Bunny	0.7	0.0167	0.258	0.082	**0.007**
	0.5	**0.0392**	0.420	0.169	0.057
	0.3	**0.0777**	0.533	0.283	0.094
Airfoil	0.7	0.103	0.279	0.184	**0.102**
	0.5	**0.105**	0.568	0.364	0.336
	0.3	**0.117**	1.979	0.876	0.782
Yeast	0.7	**0.007**	0.291	0.024	0.113
	0.5	**0.011**	1.080	0.133	0.398
	0.3	**0.03**	3.482	0.458	2.073
Minnesota	0.7	**0.0577**	0.357	0.114	0.118
	0.5	**0.0838**	0.996	0.382	0.457
	0.3	**0.0958**	3.423	1.572	2.073

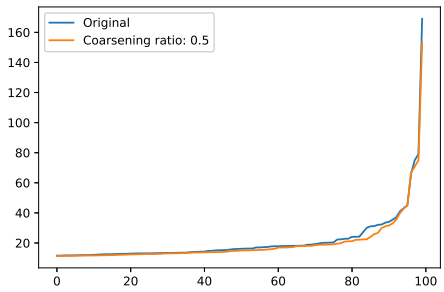
Table 9: This table summarizes the REE values obtained by FGC (proposed), GOREN(HEM), GOREN(LVN) and GOREN(LVE) on different coarsening ratios (r) for real graph datasets. It is evident that FGC (proposed) outperforms state-of-the-art methods significantly.

B.2. Spectral similarity

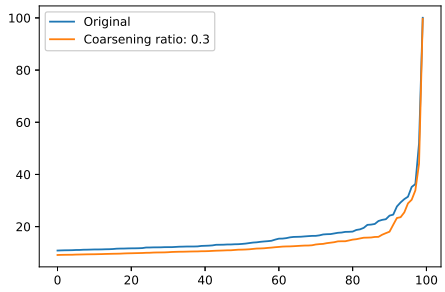
We present eigen value plots on different coarsening ratios (r) to visualize the spectral similarity of $\mathcal{G}$ and $\mathcal{G}_c$.



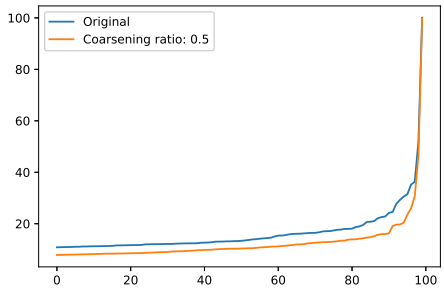
(a) Cora at r=0.3



(b) Cora at r=0.5

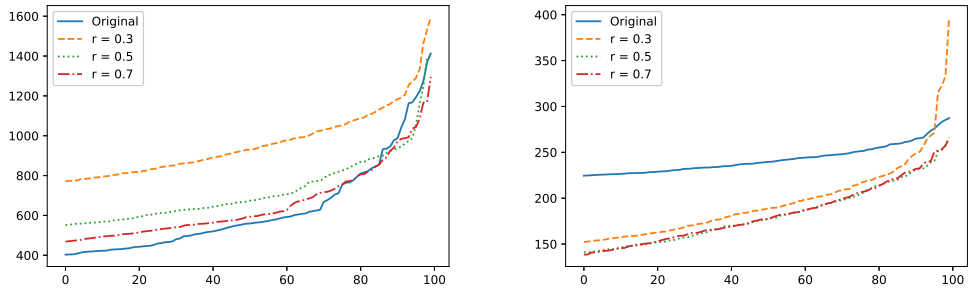


(c) Citeseer at r=0.3



(d) Citeseer at r=0.5

Figure 5: This figure plots top-100 eigen values of Laplacian matrices of original and coarsened graphs for cora (a and b) and citeseer (c and d).

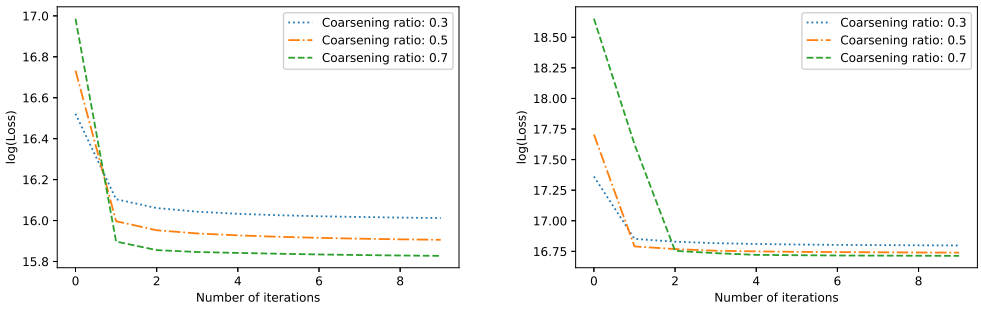


(a) BA (b) RGG

Figure 6: This figure plots top-100 eigen values for (a) BA and (b) RGG of original and coarsened graphs.

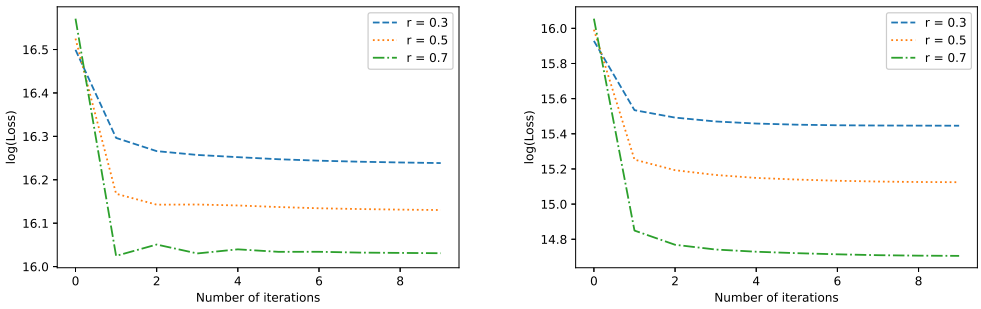
B.3. Loss Curves

Here we compute the loss curve for 10 iterations and in each iteration, C is updated 100 times having a learning rate $\frac{1}{k}$ on real and synthetic datasets to experimentally show the convergence of the proposed FGC algorithm. The figures below for different coarsening ratios demonstrate that the proposed FGC algorithm is convergent.



(a) Cora (b) Citeseer

Figure 7: This figure shows loss curves for (a) Cora (b) Citeseer on different coarsening ratios.

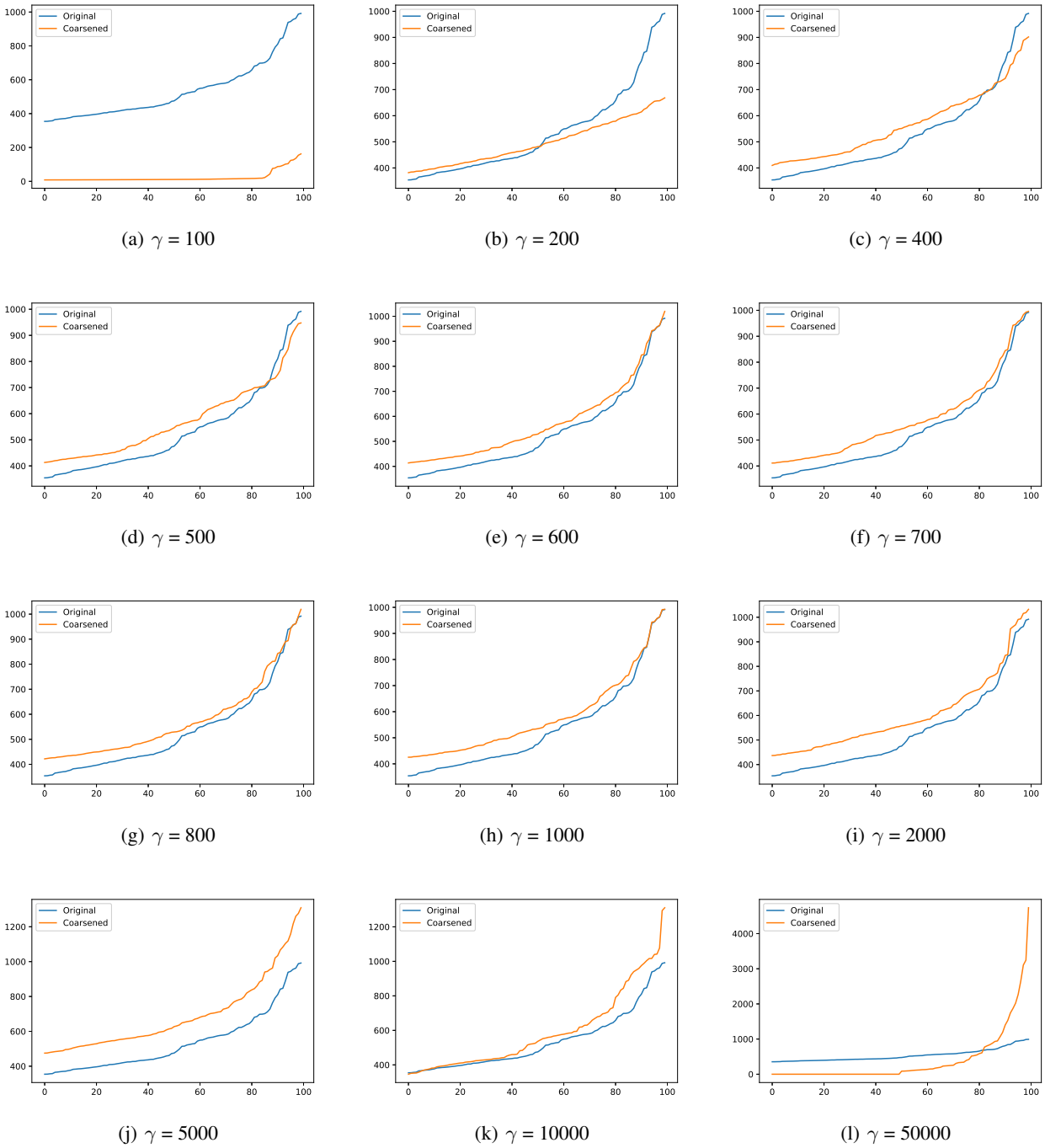


(a) WS (b) BA

Figure 8: This figure shows loss curves for (a) WS and (b) BA on different coarsening ratios.

B.4. Effects of Hyperparameters on FGC algorithm:

The FGC algorithm has 3 hyperparameters:(i) γ for ensuring the coarsen graph is connected, (ii) α to learn $\tilde{X}$ correctly (iii) λ to enforce sparsity and orthogonality on loading matrix C. From Figures 5,6, and 7, it is observed that the algorithm is not sensitive to the hyperparameters $(\lambda, \gamma, \alpha)$ any moderate value of can be used for the FGC algorithm.



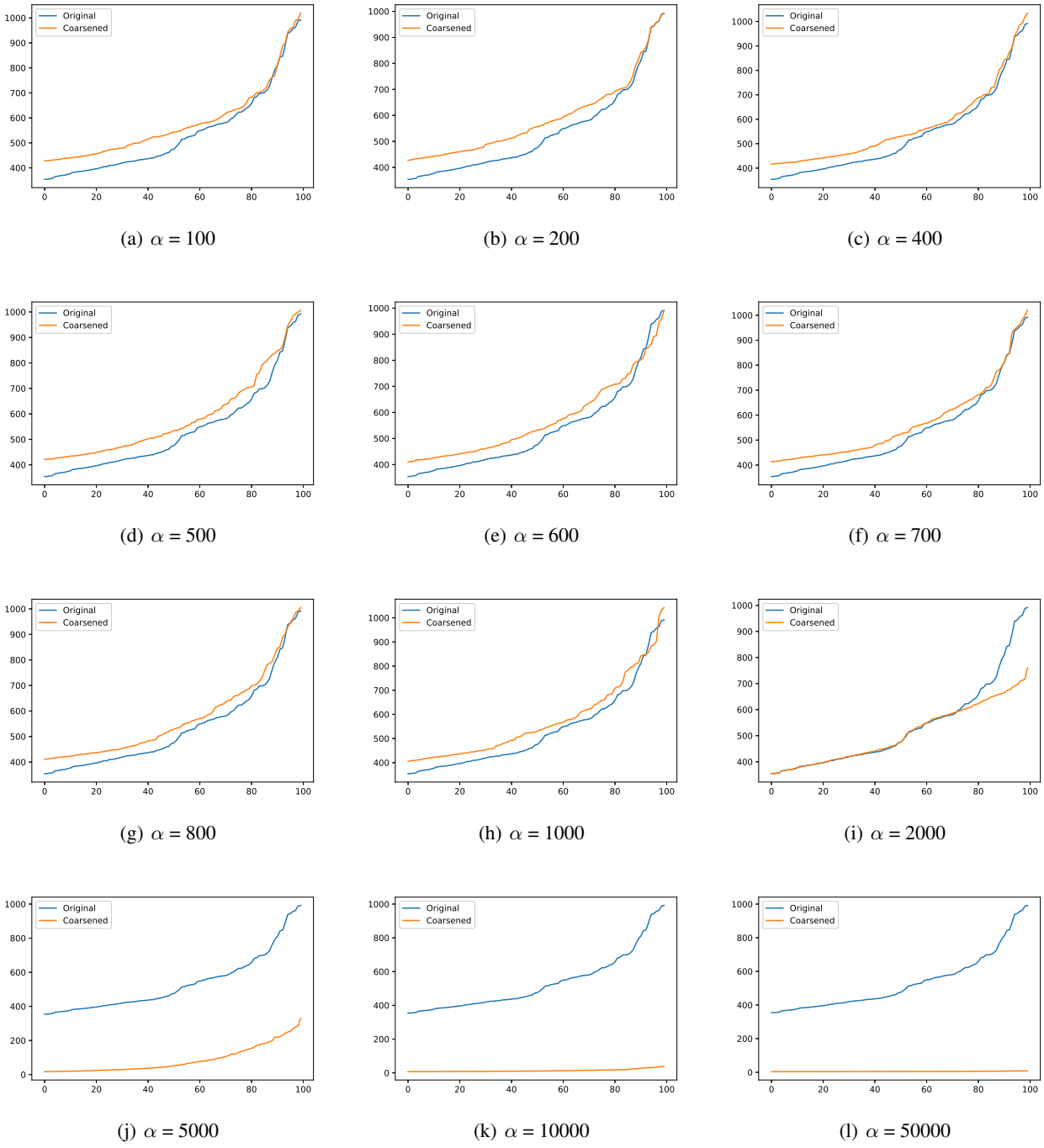
(a) γ = 100 (b) γ = 200 (c) γ = 400

(d) γ = 500 (e) γ = 600 (f) γ = 700

(g) γ = 800 (h) γ = 1000 (i) γ = 2000

(j) γ = 5000 (k) γ = 10000 (l) γ = 50000

Figure 9: Fig.(a-l) shows the eigen value plot of original graph and coarsened graph obtained by FGC using hyperparameters $\alpha = 500$, $\lambda = 1000$ and varying γ in between (100-50000). It is evident that for a moderated γ i.e between 200 to 2000, the REE is almost similar and our algorithm is consistent.



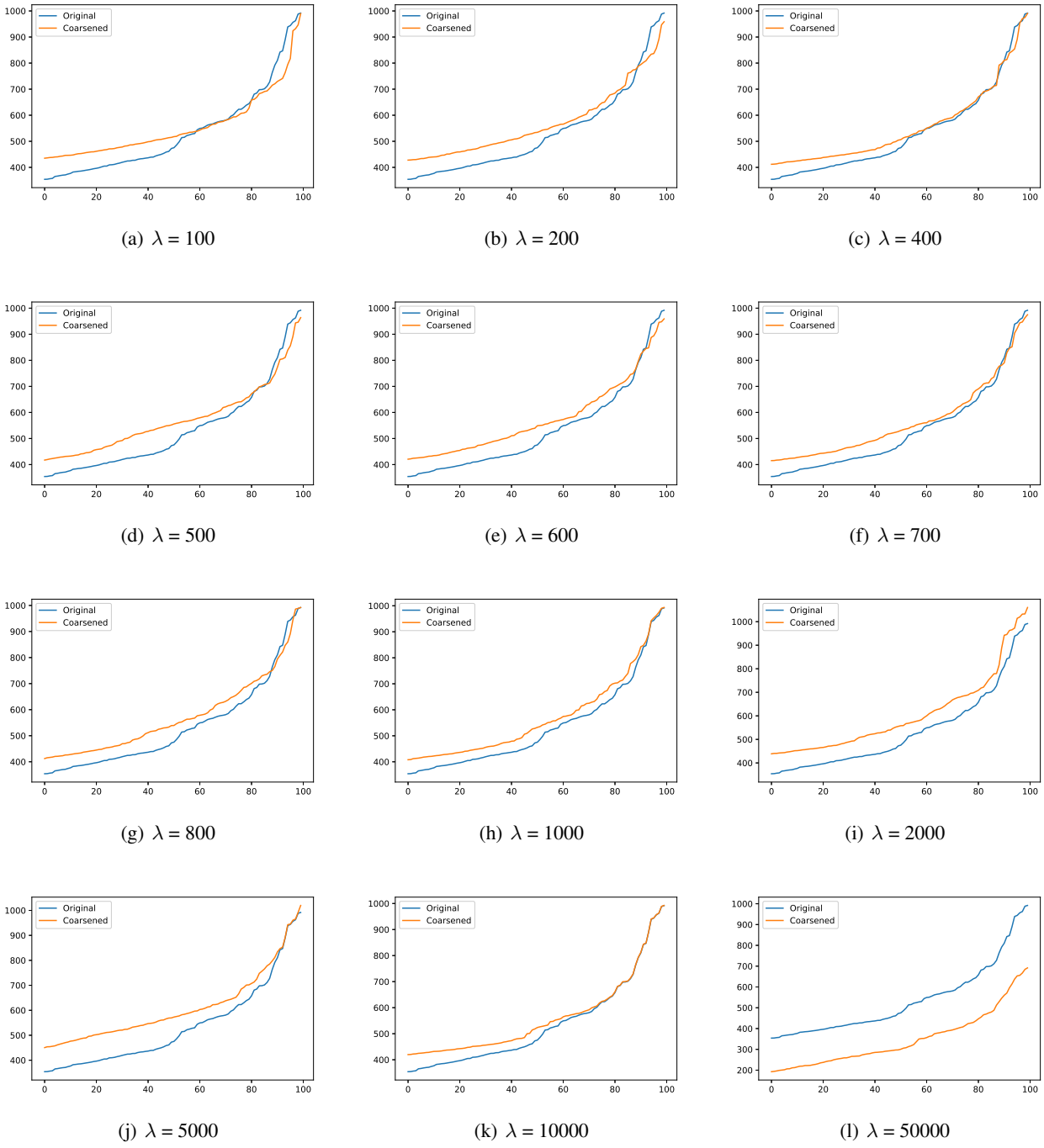
(a) $\alpha = 100$ (b) $\alpha = 200$ (c) $\alpha = 400$

(d) $\alpha = 500$ (e) $\alpha = 600$ (f) $\alpha = 700$

(g) $\alpha = 800$ (h) $\alpha = 1000$ (i) $\alpha = 2000$

(j) $\alpha = 5000$ (k) $\alpha = 10000$ (l) $\alpha = 50000$

Figure 10: Fig.(a-l) shows the eigen value plot of original graph and coarsened graph obtained by FGC using hyperparameters $\lambda = 1000$, $\gamma = 600$ and varying α in between (100-50000). It is evident that for a moderated α i.e between 100 to 2000, the REE is almost similar and our algorithm is consistent.



(a) $\lambda = 100$ (b) $\lambda = 200$ (c) $\lambda = 400$

(d) $\lambda = 500$ (e) $\lambda = 600$ (f) $\lambda = 700$

(g) $\lambda = 800$ (h) $\lambda = 1000$ (i) $\lambda = 2000$

(j) $\lambda = 5000$ (k) $\lambda = 10000$ (l) $\lambda = 50000$

Figure 11: Fig.(a-l) shows the eigen value plot of original graph and coarsened graph obtained by FGC using hyperparameters $\alpha = 500$, $\gamma = 1000$ and varying λ in between (100-50000). It is evident that for a moderated λ i.e between 100 to 2000, the REE is almost similar and our algorithm is consistent.

B.5. Clustering on Polblogs dataset

The input is a political blogs consisting of 1490 nodes, the goal is to classify the nodes into two groups. For the FGC algorithm, the feature matrix X of size 1490×5000 is generated by sampling from $X \sim \mathcal{N}(\mathbf{0}, L^{\dagger})$, where L is the Laplacian matrix of the given network. The FGC algorithm and Graclus correctly classifies 1250 and 829 nodes respectively.

However, the performance of LVN and spectral clustering are not competent. The FGC algorithm outperperforms and it also demonstrates that the features may help in improving the graph-based task, and for some cases like the one presented here the features can also be artificially generated governed by the smoothness and homophily properties.